UNIVERSITY OF CANTERBURY

DOCTORAL THESIS

Mobile Augmented Reality: Free-hand Gesture-based Interaction

Author:
Huidong BAI

Supervisor:
Prof. Mukundan RAMAKRISHNAN
Prof. Mark BILLINGHURST

A thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy

in the

The Human Interface Technology Laboratory New Zealand
Department of Computer Science and Software Engineering

December 31, 2016

Declaration of Authorship

I, Huidong BAI, declare that this thesis titled, "Mobile Augmented Reality: Free-hand Gesture-based Interaction" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date: December 31, 2016

"Good bye, old friend. May the Force be with you."

Obi-Wan Kenobi said to Anakin Skywalker, before traveling to Utapau.

University of Canterbury

Abstract

Department of Computer Science and Software Engineering

Doctor of Philosophy

Mobile Augmented Reality:
Free-hand Gesture-based Interaction

by Huidong BAI

The primary goal of this thesis is to design, implement and evaluate novel interaction techniques for enhancing Augmented Reality (AR) on mobile platforms. The motivation for this research comes from the need for more efficient and instinctive interaction approaches for mobile AR systems. With a mobile AR framework including natural feature tracking and natural gesture interfaces developed on the mobile and wearable devices, we discuss advantages and limitations of free-hand gesture for mobile AR interaction, and evaluate all our designs with users on common tasks of AR manipulations.

In this thesis, we apply typical Human-Computer Interaction (HCI) methodologies to tackle this discussion. In particular, we developed two types of markerless gesture-based input technologies in self-contained and client-server design on handheld mobile AR implementations, and one type of multi-modal interface with one wearable mobile AR implementation. We studied these interfaces separately with corresponding user studies on our mobile AR systems to analyze the usability of our proposed free-hand gesture-based interfaces, and to investigate how the interaction can be improved as well.

The results from our evaluations show that natural gesture interaction offers intuitive interaction metaphors for manipulating mobile AR scenes while being as easy to learn and use as the traditional touch-based methods. Although it is not very suitable for long periods of use because of the physical stress, the free-hand gesture-based interaction provides a more fun and engaging user experience. Meanwhile, its performance can be improved by combining the touch-based input method for a more precise operation in task refinement. Our contribution supports interface designers in making informed design decisions for mobile AR systems with free-hand gesture-based interfaces.

Acknowledgements

I am indebted to many people for their help in making this thesis possible. At this stage I wish to acknowledge and thank them for their support.

First of all, I thank my supervisor Prof. Mukundan Ramakrishnan for his constant support, believe and guidance throughout this work. Mukundan has always been positive and encouraging, and let me freedom to pursue my research goals, and I feel honoured to have had him as a mentor. I also thank my co-supervisor Prof. Mark Billinghurst. Mark's dedication to his work and the students is such an inspiration, and I am proud to have the opportunity to work with him, and learn many great things about being an academic from him as well. I also thank Dr Gun Lee, his assistance and advice has been invaluable, and I cannot express enough gratitude for the time he have invested in helping me during this research. I would also like to thank the examiners of this thesis for taking the time to read and evaluate my work.

I cannot express enough gratitude to my family, who have always supported and believed in me. To my parents and my brother, thank you so much for the patience and love all the time.

I would not have been able to complete this thesis without the support of the staff and students at the HIT Lab NZ, who always provide me a positive and lovely working environment. Thank you Thammathip, David, Hyungon, Alaeddin and Greg for all the great time and memories during my study.

I would like to say a special thank you to Lei. It is always nice to collaborate with him on interesting topics and to travel around for conferences and projects.

Contents

List of Figures

List of Tables

List of Abbreviations

AR	**A**ugmented **R**eality
CCS	**C**amera **C**oordinate **S**ystem
DOF	**D**egrees **O**f **F**reedom
FPS	**F**rames **P**er **S**econd
HMD	**H**ead **M**ounted **D**isplay
HPU	**H**olographic **P**rocessing **U**nit
HCI	**H**uman-**C**omputer **I**nteraction
IMU	**I**nertial **M**easurement **U**nit
NFT	**N**atural **F**eature **T**racking
OS	**O**perating **S**ystems
PTAM	**P**arallel **T**racking **A**nd **M**apping
PDA	**P**ersonal **D**igital **A**ssistants
PCA	**P**rincipal **C**omponent **A**nalysis
SLAM	**S**imultaneous **L**ocalization **A**nd **M**apping
TUI	**T**angible **U**ser **I**nterfaces
UMPC	**U**ltra **M**obile **PC**
UDP	**U**ser **D**atagram **P**rotocol
UI	**U**ser **I**nterface
WCS	**W**orld **C**oordinate **S**ystem

Dedicated to my dear parents.
Thank you for all of your support along the way.

Chapter 1

Introduction

Augmented Reality (AR) allows a user to see and interact with virtual imagery overlaid on his or her surrounding real environment. Using Azuma's widely accepted definition [5], AR systems have three critical characteristics; 1) they combine real and virtual images, 2) they are interactive in real time, and 3) the virtual content is registered in 3D. From this definition, it is clear that being able to interact in real time with the virtual content being shown is a key requirement.

Over the last decade mobile AR systems using smartphones and tablets have become widespread, and smartglasses running mobile AR experiences have also become very popular (see for example Figure 1.1). Following Azuma's definition, these systems need to provide ways to interact with the virtual content they are showing over the view of the real world. The first mobile AR systems supported simple button or touch screen input [29]. However, as the computational resources and sensing capabilities of mobile platforms have grown, they are becoming increasingly suitable for natural interfaces for mobile AR, such as using device motion or natural gestures.

FIGURE 1.1: Mobile AR on different platforms. Above: mobile AR in the handheld form (Source: Vuforia); Below: mobile AR in the smartglass form (Source: ODG).

1.1 A Trend Towards Mobile AR Systems

Hardware and System

The trend towards mobile AR is clear in the latest academic AR research and commercial AR products. For example, the Microsoft HoloLens[1] is the latest cutting-edge example demonstrating this trend from a few important perspectives. The HoloLens (Figure 1.2) shows the trend towards miniaturization of mobile AR hardware platforms, which started from bulky backpacks [29] [45], and then moved to handheld tablets [93] [111] and mobile phones [39] [110], and finally became current compact head-mounted smartglasses with self-contained computing, image sensing, see-through display modules and built-in battery power.

FIGURE 1.2: Microsoft HoloLens, one of the latest cutting-edge wearable AR devices (Source: Microsoft).

This all-in-one AR device not only has the equivalent CPU and RAM as a standard desktop computer, but also has its own additional Holographic Processing Unit (HPU) for specific AR related computer vision tasks. The HPU, which processes and integrates terabytes of information in real time from the HoloLens's sensor cluster, handling tasks such as spatial mapping, gesture recognition, and voice and speech recognition, and achieving stable markerless inside-out tracking. This is accomplished by using retrieved data from multiple RGB cameras and depth sensors. In addition, the HoloLens is equipped with a pair of lightweight see-through display modules calibrated for final AR content overlay.

Along with the evolution of hardware platforms, the corresponding software is also changing towards agile Operating Systems (OS), such as Android[2], iOS or embedded Linux, instead of relying on the overloaded OS of desktop computers. AR research and products based on embedded computing systems are increasing dramatically and becoming mainstream.

[1] http://www.microsoft.com/microsoft-hololens/
[2] http://www.android.com/

These operating systems are designed in a modular architecture and can be easily customized for AR without using redundant modules, which will lead to better control of function implementation, vastly reducing the demand of hardware resources while keeping a more efficient power consumption rate, and enabling better mobile AR framework design.

Tracking

Built upon the low-level system hardware and software, visual tracking and interaction technologies are the middle-ware of mobile AR core components. Marker-based tracking methods were initially widely used, because advanced tracking solutions like the natural feature based tracking were too computationally expensive to run on the early stage mobile AR hardware.

The original marker-based methods used template matching of the 2D image surrounded by a black border after quantizing the image to a matrix of programmer-specified resolution [53], and then improved the performance by introducing binary marker patterns [110]. With the development of mobile chips with more CPU and GPU cores to speed up the signal processing, markerless AR tracking algorithms could be eventually ported to mobile AR platforms, reaching real-time performance with sufficient accuracy. The markerless methods were based on natural features initially to achieve the tracking of pre-trained textured planer targets [112], and then adopted a Simultaneous Localization and Mapping (SLAM) approach to extend the tracking environment to an unknown scene, requiring no target training procedure in advance [58]. Markerless tracking even updated the tracking procedure from typical spare SLAM [82] to dense SLAM [27] to remove the natural feature or texture dependency. Meanwhile, different sensors were also integrated into the tracking system especially for outdoor environments. Figure 1.3 shows the trends in terms of the visual tracking technology of mobile AR.

Interaction

The early AR interaction methods were limited to traditional tools such as keyboard, or mouse, or handheld controllers, pairing with bulky AR systems running on desktop computers. Mobile AR interfaces first used device-centric design for interaction, such as using a keypad to complete 3D manipulations [40]. When touch screens began to be integrated into smartphones, multi-touch input provided a more easy and efficient way for users to work with augmented virtual content, and became the mainstream method for interacting with smartphone based AR applications. Mobile devices have also added more integrated sensors, which has also been used for rich AR interaction, such as camera based 6 Degrees of Freedom (DOF) operation [80], or

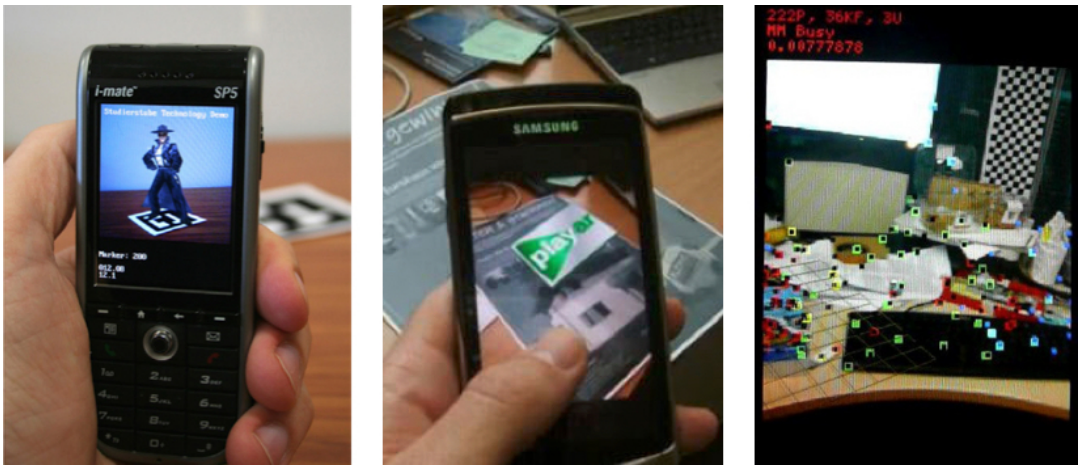

FIGURE 1.3: The visual tracking technology of mobile AR is changing from the marker-based method [110] [107] to the markerless one [59].

gyroscope based virtual object rotation. Figure 1.4 shows the trends in interaction methods for mobile AR.

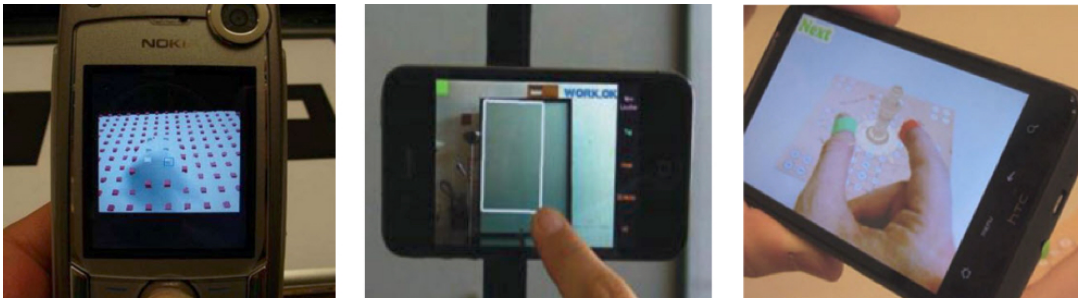

FIGURE 1.4: The interaction technology of mobile AR is changing from the device-centric method [38] [56] to the user-centric natural method [49].

For wearable computers such as the Google Glass[3], a built-in touch-bar is used for most input operations via clicking, tabbing or sliding. However, these methods require one-handed operation, which has certain limitations for bimanual tasks, and may also encounter some social acceptance issues as well. In contrast, voice or blinking input can free the user's two hands for AR interactions. This type of user-centric interface is being considered nowadays for mobile AR and especially wearable AR systems. Natural gesture based interaction is one of the most important methods being researched, while gaze based interfaces are also being investigated for head-worn AR systems.

Modern mobile AR systems use multi-modal interaction more frequently, offering flexible input options for different scenarios. For example, Microsoft HoloLens not only supports basic gesture-based interaction (with a clicker as an alternative

[3] http://www.google.com/glass/start/

option), but also the voice commands and synchronous head pointing, and a user can manipulate virtual objects with the help of different sensors like environment cameras, depth sensors and Inertial Measurement Units (IMU), etc.

As mobile AR systems become more mature and reliable, the application fields they are being used for are also changing from education and gaming to serious tasks like medical treatment, warehousing, manufacturing, construction and remote collaboration [102]. For instance, a wearable AR solution has been developed using the HoloLens for elevator maintenance that allows service technicians to visualize and identify problems with elevators ahead of a job, and have remote, hands-free access to technical and expert information when onsite (see for example in Figure 1.5) - all resulting in significant savings in time and stress[4].

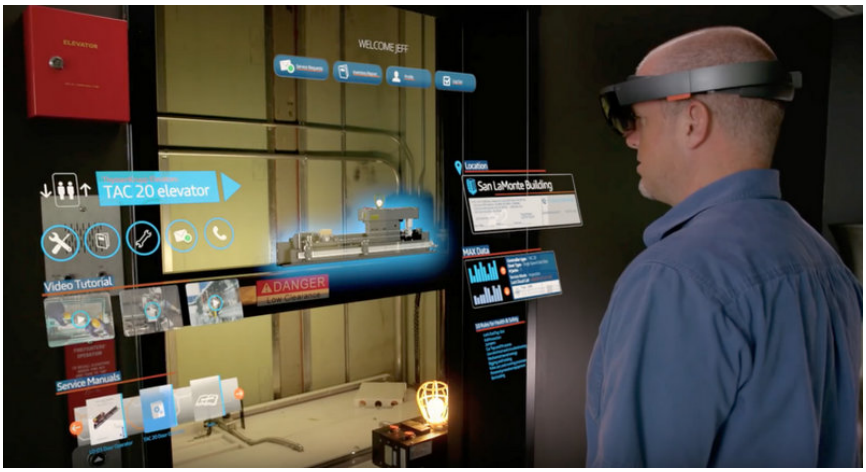

FIGURE 1.5: A wearable AR solution with Microsoft HoloLens for elevator maintenance (Source: Microsoft).

In summary, mobile AR devices are trending towards a compact head-worn AR smart glass form factor, working with an agile embedding OS to provide hybrid markerless tracking and multi-modal natural interaction. However there is considerable research that still needs to be conducted on interaction methods before the ideal mobile AR experience is achieved. Making contributions towards this is the goal of this thesis research.

1.2 Problem Statement

Mobile AR has a huge potential for providing intuitive information and direct assistance for our daily lives, and there has been significant research and commercial interest in this field. However, there are still a large number of challenging research

[4] http://blogs.windows.com/devices/2016/09/15/microsoft-hololens-enables-thyssenkrupp-to-transform-the-global-elevator-industry/

questions that need to be addressed before the relevant mobile AR technologies can be applied more widely.

Most mobile AR systems choose mobile or wearable devices as the basic platform. Unfortunately, the state of the art interaction solutions for mobile devices are barely suitable for mobile AR systems, not only because of the different characteristics of mobile AR scenarios, but also the specificity of mobile AR interactive mode. In specific, mobile AR systems augment digital content in spatial physical environment instead of flat mobile screen, which require the user to solve the input dimension issue with the help of novel interfaces.

AR technology often involves manipulating virtual images, but using a mobile AR system is also very different from traditional desktop based AR interfaces. In a typical mobile phone based AR system, the virtual content appears overlaid on the real 3D physical world, while the user's input is confined to the limited surface of devices, and there is a direct connection between the display (output) and keypad/touch screen (input). Pressing the mobile touch screen while pointing the device at a tracking image can result in camera shake that reduces the AR tracking accuracy [70], and can also cause finger occlusion of on-screen content.

Another recent development in mobile AR is the emergence of lightweight, head-mounted wearable devices that incorporate sensing and display technologies. Although these head worn mobile AR devices offer hands free interaction, their input methods also have substantial limitations. For example, Google Glass supports touch pad, button or speech input, but these methods barely enable complicated mobile AR interaction. When considering interacting with 3D image content instead of an image from a single viewpoint, conventional touch-based interaction techniques may not be as useful. For example, it can be very time consuming and uncomfortable for users to conduct 3D operations via the default input methods [8].

Vision-based natural gesture interaction provides a possible alternative solution that could provide a more intuitive user experience for 3D manipulations and enhances human performance for basic mobile AR tasks. However, due to the limited computing power of mobile CPUs and incompatibilities of mobile operating systems, the gesture detection algorithms that run smoothly on PCs barely work on mobile devices. Low quality cameras and slow graphics hardware of mobile devices dramatically reduce the performance of the final AR applications. For example, cameras on mobile platforms have a narrow field-of-view which is not suitable for robust gesture movement tracking; they have long exposure times which results in significant motion blur; and finally, they use a rolling shutter which causes severe smearing effects.

Similarly, mobile GPUs have fewer cores for complicated virtual sense rendering, resulting in a lower interactive framerate and poorer quality graphics.

A large body of previous work has mainly discussed device-centric interfaces, while only a few researchers apply user-centric methods like natural gestures to mobile AR systems. Previous research has mostly been from a technical focus to mitigate the impact of the device's computing deficiencies, as a demonstration showcase application, or to evaluate technological improvements, while usability issues are only marginally considered.

1.3 Research Questions

This thesis takes a Human Computer Interaction (HCI) perspective on free-hand gesture-based natural interaction for handheld or head-worn mobile AR systems by conducting the user interface design and the usability evaluation to target the following research questions that have not been thoroughly addressed yet:

- Q1: What are the advantages of natural gesture-based interfaces compared to touch-based ones in mobile AR environments?
- Q2: What are the limitations of natural gesture-based interfaces compared to touch-based ones in mobile AR environments?
- Q3: How can we complement the gesture-based input with other methods, like touch-based methods, to build an effective multi-modal interface?

Our basic research hypothesis is that natural gesture-based interaction is more natural and intuitive for mobile AR interaction than the popular touch-based interfaces, while being as easy to learn and use. We also assert that gesture-based interaction can enhance a mobile AR systems' usability if the limitations are properly addressed by supporting multi-modal interaction, such as seamlessly integrating touch input for accurate operations on a small working scale.

In this thesis, we test this hypothesis through standard HCI methodologies: we first select representative tasks and scenarios for mobile AR systems, and then design natural interfaces to support the selected tasks, in which we explore 2D and 3D free-hand gesture-based interaction techniques. We then conduct user evaluations to gain qualitative and quantitative insight into the usability of our interfaces, and further understand the conditions in which our proposed techniques can effectively enhance mobile AR interaction. We finally use the results from our evaluations to

reflect on our interface designs by presenting sample applications showing how these methods can be applied.

1.4 Contributions

This thesis' main contribution is developing the following three types of free-hand gesture-based interaction techniques for mobile AR applications, and the corresponding user studies that validate these techniques:

1. Free-hand Gesture-based Interaction with Self-contained Design

We present two types of gesture-based interaction that run fully self-contained on a mobile device. First, we develop a 2D fingertip-based interaction for a mobile AR system running on a smart phone. Mobile AR with handheld devices using touch interfaces requires users to hold the device with one hand and touch the screen with the other, while aiming the mobile camera at an AR target to maintain visual tracking. However, pressing the touch screen while pointing the device at an AR tracking image can result in camera motion that has a serious negative effect on the AR tracking accuracy, and overall user experience. We investigate two different approaches to overcome this problem; (1) mid-air 2D gesture input and (2) freeze view touch, and we also compared these two methods with conventional touch input in a formal user study.

Second, we explore 3D fingertip-based interaction for a mobile AR system working on a tablet with an RGB-Depth camera attached via a USB connection. Since existing touch-based input and most gesture interaction techniques only use two degrees of freedom without the depth dimension, it is difficult to map two-dimensional input into a 3D spatial AR environment. To overcome this, we present a novel method that is based on identifying the 3D positions and movements of the user's fingertips using depth data, and mapping these gestures onto corresponding manipulations of the virtual objects in the AR scene. We conducted a user study to evaluate this method by comparing it with a common touch-based interface under different AR scenarios.

2. Free-hand Gesture-based Interaction with a Client-Server Framework

More natural 3D gesture based interaction can be prototyped with a client-server framework that connects a mobile device to a desktop computer for gesture tracking. For the client-server design, we also present two types of interactions. We first develop a markerless fingertip-based 3D interaction method within a client-server framework in a small workspace, using visual tracking based AR and freehand gesture-based interaction detected by a depth camera, and then extend it by supporting full hand

skeleton detection for more accurate and rich AR manipulations. We evaluated this prototype in a user study comparing 3D gesture input methods with traditional touch-based techniques, using canonical manipulation tasks that are common in AR scenarios.

Based on this framework, we studied hand gesture-based interfaces for real-time 3D AR modeling and animation on handheld mobile devices, and described four manipulation methods with an emphasis on the most common modeling tasks in an AR environment. We also presented an augmented exhibition podium that supports natural free-hand 3D interaction for visitors using their own mobile phones or smartglasses. Visitors can point the camera of their mobile phones or Smart Glasses at the podium to see AR content overlaid on a physical exhibit, and can also use their free-hand gestures to interact with the AR content. We conducted a pilot user study to test the usability of this system.

3. Multi-Modal Interaction with Complementary Free-hand Gestures

We further report a multi-modal interaction method to complement touch input by adding free-hand gesture input technology on a wearable AR smart glass, with which the user can easily combine low-resolution hand gestures in 3D with high-resolution touch input. We show how this could enhance task completion in a wearable AR interface and present early pilot study results.

In summary, the results presented in this thesis deepen the understanding of how free-hand gestures can be effectively integrated into mobile AR systems. The results from this thesis could help support interface designers in making informed design decisions. Furthermore, the user evaluations presented in this thesis provides an overview of possible methodologies for evaluating mobile AR's natural interfaces in handheld or wearable systems.

The organization of rest of dissertation is described here. Chapter 2 covers the background research with the focus on mobile AR platforms and user interface development. Development of free-hand natural gesture interface with a self-contained design and a client-server framework and corresponding evaluations are described in Chapter 3 and Chapter 4 respectively, and three applications based on Chapter 4's framework design are explored in Chapter 5. Conclusion and future research are explained in Chapter 6.

1.5 Selected Publications

The list of selected peer-reviewed publications below gives an overview of the scientific activities and the collaborations which occurred during the work of this thesis.

In the following publications, the author was mostly involved in the interface design and final evaluations of the study, making minimal contributions to the implementation and data analysis. For the book chapters and survey papers, the author's major contributions cover the literature review section of mobile AR systems and their interaction methods.

- Lei Gao et al. "An Oriented Point-cloud View for MR Remote Collaboration". In: *SIGGRAPH ASIA 2016 Mobile Graphics and Interactive Applications*. SA '16. Macau, China: ACM, 2016, 8:1–8:4
- Alaeddin Nassani et al. "Tag It!: AR Annotation Using Wearable Sensors". In: *SIGGRAPH Asia 2015 Mobile Graphics and Interactive Applications*. SA '15. Kobe, Japan: ACM, 2015, 12:1–12:4
- Mark Billinghurst, Tham Piumsomboon, and Huidong Bai. "Hands in Space: Gesture Interaction with Augmented-Reality Interfaces". In: *IEEE Computer Graphics and Applications* 34.1 (2014), pp. 77–80
- Mark Billinghurst et al. *Developing Handheld Augmented Reality Interfaces*. New York, NY, USA: Oxford University Press, 2014, pp. 615–635
- Cik Suhaimi Yusof et al. "A Review of 3D Gesture Interaction for Handheld Augmented Reality". In: *In Proceedings of the 3rd International Conference on Interactive Digital Media*. ICIDM '14. Sabah, Malaysia, 2014
- Gun A. Lee, Huidong Bai, and Mark Billinghurst. "Automatic Zooming Interface for Tangible Augmented Reality Applications". In: *Proceedings of the 11th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry*. VRCAI '12. Singapore, Singapore: ACM, 2012, pp. 9–12

The following publications have a strong HCI focus and form the main contribution of this thesis. The author was a main contributor at all stages of the work – forming the design ideas, designing and implementing the interfaces, designing and conducting user evaluations, and analyzing and discussing the results.

Chapter 3 Free-hand Gesture-based Interaction with Self-contained Design

Section 3.1 2D Markerless Fingertip-based Interaction
Section 3.2 Evaluation

- Huidong Bai, Gun A. Lee, and Mark Billinghurst. "Freeze View Touch and Finger Gesture Based Interaction Methods for Handheld Augmented Reality

Interfaces". In: *Proceedings of the 27th Conference on Image and Vision Computing New Zealand*. IVCNZ '12. Dunedin, New Zealand: ACM, Nov. 2012, pp. 126–131

Section 3.3 3D Markerless Fingertip-based Interaction

- Huidong Bai et al. "Free-hand Interaction for Handheld Augmented Reality Using an RGB-depth Camera". In: *SIGGRAPH Asia 2013 Symposium on Mobile Graphics and Interactive Applications*. SA MGIA '13. Hong Kong: ACM, Nov. 2013, 22:1–22:4

Section 3.4 Evaluation

- Huidong Bai et al. "Work-in-progress: Markerless 3D Gesture-based Interaction for Handheld Augmented Reality Interfaces". In: *Proceedings of the 16th IEEE International Symposium on Mixed and Augmented Reality*. ISMAR '13. Adelaide, South Australia, Australia: IEEE Computer Society, Sept. 2013, pp. 1–6

Chapter 4 Free-hand Gesture-based Interaction with Client-Server Framework

Section 4.1 Finger-based 3D Interaction within A Client-Server Framework

- Huidong Bai, Lei Gao, and Mark Billinghurst. "Poster: Markerless Fingertip-based 3D Interaction for Handheld Augmented Reality in a Small Workspace". In: *Proceedings of the 8th IEEE International Symposium on 3D User Interfaces*. 3DUI '13. Orlando, FL, USA: IEEE Computer Society, Mar. 2013, pp. 129–130

Section 4.2 Skeleton-based 3D Interaction within A Client-Server Framework
Section 4.3 User Study

- Huidong Bai et al. "3D Gesture Interaction for Handheld Augmented Reality". In: *Proceedings of the 7th ACM SIGGRAPH Asia 2014 Symposium on Mobile Graphics and Interactive Applications*. SA MGIA '14. Shenzhen, China: ACM, Nov. 2014, 7:1–7:6

Chapter 5 Prototype Applications

Section 5.2 Multi-modal Interaction

- Huidong Bai, Gun A. Lee, and Mark Billinghurst. "Using 3D Hand Gestures and Touch Input for Wearable AR Interaction". In: *Extended Abstracts on ACM CHI 2014 Conference on Human Factors in Computing Systems*. CHI EA '14. Toronto, Ontario, Canada: ACM, Apr. 2014, pp. 1321–1326

Section 5.3 Augmented Exhibition Podium

- Huidong Bai, Gun Lee, and Mark Billinghurst. "Free-hand Gesture Interfaces for an Augmented Exhibition Podium". In: *Proceedings of the Annual Meeting of the Australian Special Interest Group for Computer Human Interaction*. OzCHI '15. Melbourne, VIC, Australia: ACM, 2015, pp. 182–186

Chapter 2

Related Work

In this chapter, in Section 2.1 we first begin with a review of previous work regarding basic mobile AR platforms, from early stage Head Mounted Display (HMD) based AR, to handheld mobile AR, and recent head-worn wearable AR. In Section 2.2 we then outline recent research in mobile AR interaction from both device-centric and user-centric perspectives, and in Section 2.3 summarize the main tasks that require further investigation in the area of free-hand gesture-based interaction for mobile AR.

2.1 Mobile AR Platforms

The basic idea of AR is to seamlessly superimpose graphics, audio and other sense enhancements over a surrounding real-world environment in real time. To do this requires a combination of display, tracking and interaction technologies. Most of the original AR systems used HMDs that were tethered to fixed desktop computers. However, since early 2000 it has been possible to run AR application on handheld and mobile devices, and the platform for the mobile AR systems has vastly evolved in the past decade. In this section we review research in mobile AR in more detail.

2.1.1 Head Mounted Display AR

From Sutherland's work onwards [105], most early AR systems used a HMD to view the overlaid graphics content. However, these HMDs were tethered to fixed computers and were too cumbersome and looked too unusual to use in everyday social settings.

In the mid 1990s, as computers increased in power and decreased in size, researchers began to explore wearable and pervasive computing AR designs. This was possible due to a new kind of AR application that exploited the person's surrounding context. The Touring Machine [29] and MARS project [45] were two of the first mobile AR systems that allowed the user to freely walk around in the real world, while

carrying all necessary system hardware to add a virtual layer of information to any environment whenever desired.

The Tinmith AR system [88] also allowed users to move freely in an outdoor environment, and was controlled using a set of pinch gloves in a specialised user interface. While traditional AR systems at the time tended to passively consume information, this system allowed users to create 3D spatial information in real time, and add virtual cues to the view of the real world. Several other prototypes such as BARS [51], and mobile Studierstube [97] were used for mobile AR applications.

However, these AR systems had the common disadvantages that they were bulky, could only be used for short periods of time due to user fatigue and limited battery life, and had novel user interfaces that were difficult to learn even with some training. For example, the original MARS system weighed over 40 pounds and was built on a custom wearable PC, with GPS hardware, an inertial head tracking system and a see-through HMD (Figure 2.1). The Tinmith system used a custom glove based input device that took time to learn its basic operations. It is difficult to imagine systems this complex being used for long periods of time, or being able to be mass-produced and used by many people.

FIGURE 2.1: Side view (left) and back view (right) of MARS system [45].

In summary, early HMDs and backpack systems made mobile AR possible, but were relatively complex, expensive, fragile and heavy, rendering them unfit for large-scale deployment involving untrained users outside constrained laboratory environments.

2.1.2 Handheld AR

One of the biggest trends in AR was in moving from using HMD and desktops to handheld devices as a lightweight alternative solution. This immensely decreased the AR systems' dependence on bulky hardware.

The first handheld AR systems dates back to Rekimoto's TransVision system [93] and Fitzmaurice's Chameleon [30]. Both used a palmtop size video see-through LCD display that was tethered to a workstation and a magnetic spatial sensor to find the position and orientation of the display. TransVision augmented a real table-top with computer graphics objects. Two or more participants held a palmtop size see-through display each and looked at the world through the display. They could both see the same virtual object and then manipulate it in a collaborative face-to-face AR experience. In this way, users could share the same virtual information in the real-world environment, each looking at it from his or her own point of view. Fitzmaurice's Chameleon consisted of a flat-panel display that allowed a user to navigate through a virtual 3D space by changing the location and orientation of the palmtop in his or her hand. The display acted as a physical window into 3D virtual environments, through which a one-to-one mapping between real and virtual space was preserved.

With a similar design as TransVision, Rekimoto's NaviCam [92] used a miniature gyro sensor to determine the orientation of device, and added ID recognition capability by analysing video images from a camera to recognize coloured codes to detect a rough position of the device in the real world (Figure 2.2). This enabled the user to see augmented information about moveable objects tagged with the code.

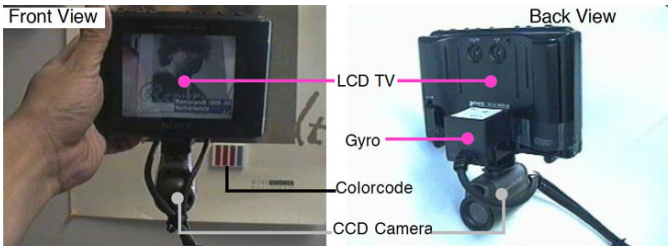

FIGURE 2.2: The outer view of NaviCam [92].

As significant computing and graphics power became available on the handheld platform, researchers naturally began to explore the use of Personal Digital Assistants (PDA) for AR applications as well. First there was work such as the AR-PDA project [33] and BatPortal [85]. AR-PDA used a PDA as a thin client that took pictures via an integrated camera and sent the video stream to a remote AR server. The server recognized objects by analysing the image and establishing the

relevant context-sensitive information, which was then added to the video stream as multimedia elements and sent back to the PDA (Figure 2.3). AR-PDA allowed the object recognition and tracking of non-moving objects, an augmentation of the scene using animated 3D objects or pictures and a personalized user interaction by touching the display with a pen. It focused on the close registration of real and virtual objects, which had been restricted to small volumes.

FIGURE 2.3: Transmission of the videostream between the AR-PDA and the server [33].

In contrast, BatPortal concentrated on allowing the AR user to roam freely within an entire building. Its "Bat" sensor system calculated the position of each Bat by measuring the time taken for ultrasonic pulses emitted by the Bats to reach receivers in known, fixed positions in the ceiling. The positions of the user's personal Bat and the Bat fixed to the handheld device were combined to form a direction vector in which the user was looking (Figure 2.4). Augmentation information could then be rendered on a PDA based on the user's location and direction of view, using the information in a sentient system's model over the WaveLAN[1] wireless link.

Neither of these systems were equipped with an inertial tracker, and so the user's position and viewing direction was only updated a few times per second, and were subject to the server's errors. However, this distributed structure was necessary as the early PDA's did not have enough capability for stand-alone mobile AR implementation.

The whole situation changed in 2003 when Wagner used a popular marker-based tracking toolkit (ARToolKit[2]) directly on an unmodified PDA with a commercial camera, and developed the first stand-alone AR system with self-tracking [111]

[1] http://www.wavelan.com/
[2] http://artoolkit.org/

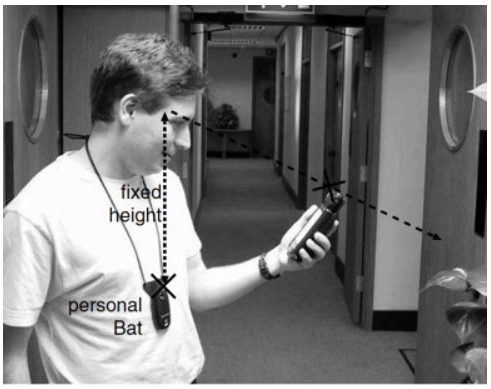

FIGURE 2.4: The direction vector calculation for the BatPortal's augmentation [85].

(Figure 2.5). ARToolKit is one of the most well-known AR tracking libraries, and uses computer vision techniques to detect squares in an image and match the interior pattern with known templates to identify the AR tracking marker and establish the main orientation and position of the camera relative to the marker.

FIGURE 2.5: The self-contained handheld AR system that can track optical markers in real time [111].

After that, the first stand-alone interactive multi-user AR application, the Invisible Train [114], was developed, running on off-the-shelf handheld devices. The Invisible Train was a simple multi-player game, in which players steered virtual trains over a real wooden miniature railroad track. The application was distributed, synchronizing state between multiple clients (up to four) through wireless networking, and a static arrangement of multiple fiducial markers in the environment enabled PDAs to perform self-tracking from a variety of different angles and distances. Figure 2.6 shows two users playing the game at the same time, as seen from the user's perspective.

Handheld systems have several advantages over traditional head-up or head-mounted configurations; a user does not have to put on any cumbersome head gear, and can easily move the device around and compare real and augmented images.

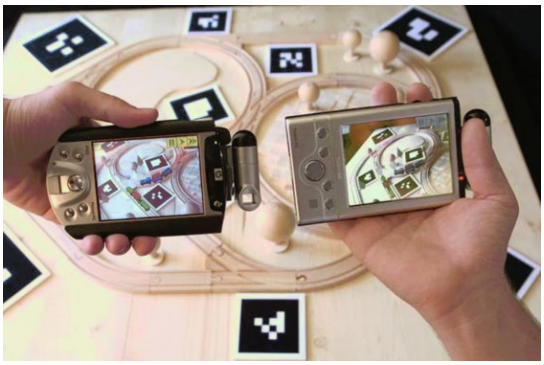

FIGURE 2.6: Two PDAs running the Invisible Train game at the same time [114].

2.1.3 Mobile AR

The current revolution in the field of mobile phones provides another excellent opportunity for AR development. The mobile phone is an ideal AR platform because new-generation phones have full-colour displays, integrated cameras and sensors, fast processors, and even dedicated 3D graphics chips.

A mobile phone can also provide convenient interaction approaches and a friendly user experience, for example, its low weight allows comfortable single-handed use. Most importantly, the mobile phone is a ubiquitous device capable of providing an AR experience to hundreds of millions of users. In the past, the cost and complexity of AR hardware prevented widespread use of AR applications, but with a mobile phone people already have the required hardware in their pocket.

Mobile AR Systems

Mobile phone based AR has followed a similar development path to handheld AR. Early phones did not have sufficient processing power to perform image processing, so researchers explored thin client approaches that could overcome the resource constraints of mobile devices by outsourcing calculations to PCs connected via a wireless connection. For example, the AR-Phone project [4] used Bluetooth to send phone camera images to a remote sever for processing and graphics overlay, taking several seconds per image. This approach suffered from low performance due to restricted bandwidth as well as the imposed infrastructure dependency, which limited scalability in the number of client devices. These server-based approaches are not real-time, and typical response times were reported to be around 10 seconds for processing a single frame.

To achieve real-time processing, stand-alone mobile AR applications were explored. For instance, TinyMotion [115] tracked the movements of cell phones in

real-time by analysing image sequences captured by the built-in camera using the optical flow method while requiring no additional sensors, but did not deliver any kind of pose estimation. Another early AR demonstration running on a cell-phone was Nokia's MARA project [52]. The system did not perform any image analysis with the on-board camera; instead using an external GPS receiver for localization, an accelerometer to provide relative orientation, and a tilt compensated magnetometer to determine heading (Figure 2.7).

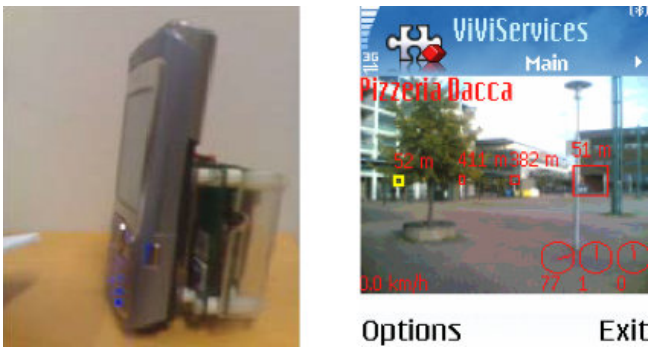

FIGURE 2.7: Mobile device equipped with additional sensor (left) and running AR see through mode (right) [52].

Henrysson and Ollila presented UMAR [41], a conceptual framework for developing ubiquitous mobile AR applications in context aware environments. They successfully ported the ARToolKit to consumer mobile phones running the Symbian[3] and brought full AR experience. Based on this technology, they presented the AR-Tennis game, the first face to face collaborative AR application running on a mobile phone [39] (Figure 2.8). Over time, more practical stand-alone mobile AR applications became real and gradually become the mainstream of AR styles.

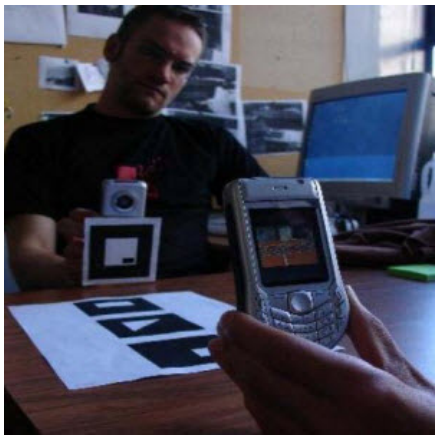

FIGURE 2.8: Playing the AR Tennis face to face [39].

[3] http://symbian.nokia.com/

Wearable devices have been recently developed that incorporate computing, sensing and display technology into a head-worn package, also becoming a promising platform for mobile AR applications. For example, Rompapas [94] demonstrated a dynamic mobile AR X-Ray system that involves two people wearing Google Glass wearable computers. One Google Glass is used to stream images and pose data to the other Glass, which uses this information for augmented X-Ray visualization. The system uses a live camera video feed, enabling the streaming user to move freely.

Mobile AR Tracking

In this thesis, we are mainly interested in user interaction, one core module of mobile AR, as our research topic. However, to set up the system for the interaction study, the very first step is to implement visual tracking, another core module of mobile AR in our research scope.

Visual tracking is the process of calculating the transformation between a known object or an unknown scene, called a trackable target, and the camera which is viewing it. This transformation is a representation of the location and orientation of the trackable target from the perspective of the camera in 3D space. The orientation is comprised of three rotations; yaw, pitch and roll, and the location is three translations; vertical, horizontal, and in depth. AR requires the transformation between the trackable target and the camera to correctly augment the real world with computer graphics. The visual tracking technology for mobile AR was initially based on marker-based methods, and later significantly improved by using markerless tracking solutions.

As we mentioned in Section 2.1.2, ARToolKit used computer-vision based template matching of a 2D image surrounded by a black border after quantizing the image to a matrix of programmer-specified resolution. This provided a popular implementation of AR visual tracking for mobile devices. ARToolKitPlus [110], a successor to the ARToolKit, was especially optimized and extended for use on mobile devices such as smartphones, PDAs and Ultra Mobile PCs (UMPC) (Figure 2.9). It introduced binary marker patterns which implicitly encoded the marker's id (with up to 4096 id markers) with built-in forward error correction. The detection of id-markers was always faster than for template-markers since no image matching was required.

Computer vision methods have also been used to implement Natural Feature Tracking (NFT) on mobile devices, where the image being tracked does not need to have a square black border around it. PhoneSIFT and PhoneFerns [112] were methods that achieved interactive frame rates of up to 20 Hz for NFT from textured planer

FIGURE 2.9: Running the ARToolKitPlus on different handheld devices with the binary marker [110].

targets on the mobile phone by successfully combining its own tracker with heavily modified state-of-the-art descriptors, namely SIFT [75] and Ferns [72]. However, to use these approaches the user needs to train the target in advance to generate the template file for corresponding tracking.

Parallel Tracking and Mapping (PTAM) [58] extended the markerless tracking environment from a pre-trained planer target into an unknown scene, such as a small AR workspace, requiring no target training procedure at all. PTAM adapted conventional SLAM algorithms [26] by splitting tracking and mapping into two separate tasks, processed in parallel threads on multiple core CPUs.

ORB-SLAM [82] further improved on the tracking range using the ORB [96], a very fast binary descriptor that is an alternative to SIFT, and provided monocular loop closing SLAM navigation within a much larger space. This method could also be used to retrieve the feature points of a 3D object or scene as a target template first in mapping and loop closing mode, and then run in localization mode only for further 3D tracking. The updated ORB-SLAM 2 [83] supported not only monocular, but stereo and RGB-Depth cameras for environment tracking in real time. Both algorithms work well on mobile phones but rely heavily on feature points for pose tracking and estimation, and so barely handle targets with a small amount of textures.

LSD-SLAM [27] [28] provided a direct (feature-less) solution that used all pixels instead of distinctive features to build a large-scale, consistent map of the environment, and to estimate accurate camera pose from a dense array of features. It was able to achieve an interactive performance on the mobile phone as well, without requiring highly textured targets for tracking and is more stable. With retrieved the semi-dense depth map for reconstruction, LSD-SLAM also offered more complicated features for mobile AR systems, such as depth occlusion and more realistic physical collision.

Omnidirectional LSD-SLAM [20], a real-time, direct monocular SLAM method for omnidirectional or wide field-of-view fisheye cameras was then developed based on the LSD-SLAM. Both its tracking (direct image alignment) and mapping (pixel-wise

distance filtering) were directly formulated for the unified omnidirectional model, which can model central imaging devices with a field of view well above 150 degree.

In addition to purely vision based approaches, sensors like IMU or GPS were commonly used to compensate for the limitations of visual tracking in hybrid tracking scenarios. For example, from Klein' research [57], a markerless visual tracker which was robust to rapid head motions was presented by combining visual measurements with those of head-worn inertial sensors. This type of sensor fusion approach allowed not only pose prediction, but also enabled the tracking of video with unprecedented levels of motion blur. For handheld AR devices, through the external fusion of these complementary sensors, accurate and robust tracking was achieved within a modest computing budget, which allowed further visual analysis of the occlusion boundaries between real and virtual objects and a marked improvement in the quality of augmentations.

Kurz and Benhimane [65] investigated how different stages in mobile AR applications could benefit from knowing the direction of gravity measured with inertial sensors. By incorporating the Gravity-Aligned Feature Descriptors in the presence of (close to) vertical surfaces and Gravity-Rectified Feature Descriptors for (close to) horizontal surfaces [66], their research greatly improved the description and matching of feature points, detection and tracking of planar templates, and the visual quality of the rendering of virtual 3D objects (Figure 2.10).

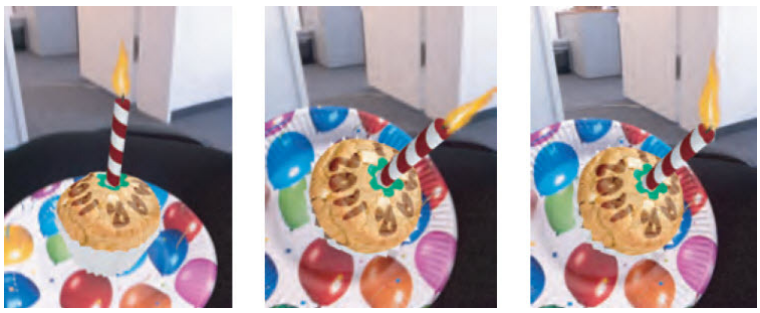

FIGURE 2.10: Aligning the candle fire with the measured gravity gives a believable impression information [66].

Wagner et al. [113] described a novel approach for the real-time creation and tracking of panoramic maps on mobile phones. They chose a cylindrical mapping to create a map of the environment based on natural features: for each frame, the camera is first registered based on features in the map, and then the map is extended with new features from viewing directions that have not been observed before. After that, they used drift-free orientation tracking which could be initialized using data from a GPS sensor and updated by comparing the camera image with the counterpart in the map. The method can be used for augmenting previously created panoramic

images, for example, an accurate mapping of the annotations to physical objects can be applied, while annotations can be tagged with the user's GPS position (Figure 2.11), stored on a server and retrieved by other users.

FIGURE 2.11: The panoramic AR system combines the real-time visual tracking and mapping with GPS data [113].

2.2 Mobile AR Interaction

As mobile AR tracking systems were being developed, there was also research being conducted on different interaction techniques for these systems. Most of these techniques were basically designed around keyboard input or the touch screen on mobile phones, or the touch panel on smart glasses, where it is easy to use stylus or touch input. However, there are other possibilities as well, like camera-based control or gesture input. We mainly reviewed related work in the following two categories from the perspective of interaction approaches:

- Device-centric interaction for mobile AR
- User-centric interaction for mobile AR

2.2.1 Device-centric Interaction Methods

With the integration of different sensors, the mobile device itself has gradually become the connection bridge linking users and 3D virtual objects in mobile AR applications: the user can indirectly manipulate the virtual content with the help of the interface supported by mobile devices. To support this, device-centric interaction methods are used.

One common technique is to use camera tracked motion of the mobile device interface. By analysing the video stream captured by the camera on the phone, it is possible to estimate the movement of the device itself, which can be used in several ways, such as providing a 6DOF manipulation, or recognizing objects to

make the phone context aware. For example, Henrysson et al. [42] attempted the first exploration of using the camera tracked motion to manipulate 3D graphical content in 6DOF to create AR scenes on mobile phones (Figure 2.12). They ported ARToolKit to Symbian, and presented a mobile AR assembly application, in which 3D objects could be manipulated by using both the movement of a camera tracked mobile phone and traditional button input. The main limitation was the tracking range as the ARToolKit square marker needed to be visible at all times, although this could be extended by using multiple markers. By arranging the scene in a tangible marker space in front of the phone they provided a means for bimanual interaction.

FIGURE 2.12: Using the camera movement to place the Lego block in 3d space [42].

Andel et al. [3] developed a novel implementation of a distributed scene graph structure for interactive collaborative mobile AR. They followed up with a user study of a furnishing application that was based on the platform for group interaction on mobile phones, in which the phone was considered to be a tangible interaction tool and was used with the phone motion to translate objects. The user study revealed that untrained users quickly adopted the technology, in an intuitive way, and could easily move the furniture but sometimes found it difficult to get the furniture in exact position they wanted.

Device motion can also be used to modify object geometry. Henrysson and Billinghurst developed a mesh editing system [38] that allowed a user to select one or more vertices in an arbitrary sized polygon by using a button on the keypad of the mobile phone, and freely translate and rotate them by translating and rotating the device itself (Figure 2.13). The phone motions were detected through using the ARToolKit computer vision tracking library. This showed how atomic actions (selection, translation, rotation) could be implemented and successfully applied in a mobile AR environment with camera tracked motion interfaces.

Chen et al. [22] introduced a client-server Chinese chess game system with mobile

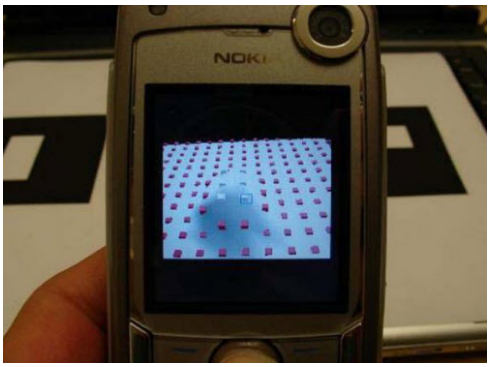

FIGURE 2.13: Using the camera tracking to edit the mesh [38].

AR technology that also demonstrated objects motion. To play this game, a user could "directly" pick up a virtual playing piece by one click of a keypad button and move it through visual tracking of the mobile phone while clicking the button again after aiming at the destination. This was similar to the way people pick-up and place pieces on a real board. This approach provided the user with a more natural human-computer interface, and a user study with the system found that the users' learning curves were greatly improved for both purposes of education and entertainment.

In-situ 3D content creation and annotating in small workspaces or unprepared environments can provide good examples of mobile AR interaction. Simon [101] developed a purely image-based mobile AR system for in-situ 3D sketching of polyhedral scenes. In this case, a video camera on the mobile phone was used as both an interaction and tracking device. For example, if a line needs to be drawn on a video image between two physical points of a real-world scene, the user can rotate the camera so that the physical points apparently move to a fixed cursor, for instance at the centre of the image; at this moment, the user can press a key to select the first point which will be tracked later. The same method can also be applied to the second point so that the line can be completed. After finishing the modeling, the built parts of the scene are immediately shown superimposed on the environment, which allows the user to verify the geometry against the physical world in real-time (Figure 2.14). The study proved that in-situ modeling interactions are possible using a mobile camera as the only input device.

Langlotz et la. [68] also developed a mobile AR system that allowed in-situ content augmentation in unprepared space. In this system, a user can use the same interaction method, camera tracking and stylus input to create new 2D or 3D content as well as replicating existing 3D objects in small working spaces or larger outdoor environments (Figure 2.15). The manipulation and authoring of the virtual content is done directly on the mobile device using an image freezing mode to support accurate

FIGURE 2.14: Screenshots of the system in use: drawing a line with two points (left) and displaying the augmented virtual content (right) [101].

operations on the device while pointing the camera to the designated spot. Once the user is in the position where he or she wants to start the authoring task, the view can be frozen by pressing a button on the touch screen. This simulates a fixed position by freezing the current camera frame and allows the user to move the device without changing the camera image. After successful completion of the selected authoring task, the user can unfreeze the view.

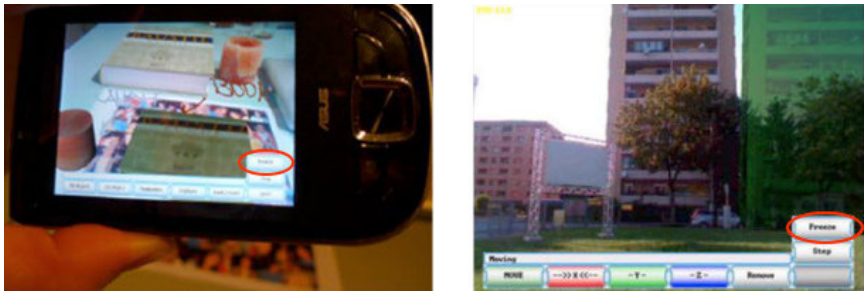

FIGURE 2.15: With the image freezing mode highlighted, the user can create augmented 3D content for small working environment (left) or large outdoors environment (right) [68].

Using the similar idea, Kim et al. [56] presented an interactive method for modeling several rooms and annotating locations with virtual content on a mobile phone. Typically, a user stands in a fixed position looking around holding the phone, and measures the dimensions of a room by selecting the edges of walls through the touch screen. The system will use this input to quickly model the room, approximated as a box. Once the room modeling is completed, the user can touch and mark-up locations with a rectangle where virtual content, like text, image and 3D models, can be overlaid (Figure 2.16). In this case, the camera-based modeling for the room extended the interaction range of the touch screen from a 2D textured plane to 3D real space.

Instead of virtually labeling the environment around the user, Liu et al. [74]

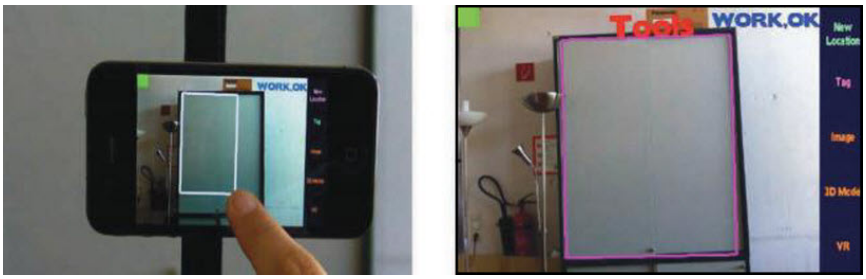

FIGURE 2.16: Selecting a rectangle target (left) and augmenting the annotation on top of it (right) [56].

explored an approach to assist operating physical devices by mapping digital information to physical objects, such as displaying virtual animated tapping footprints between the keys to illustrate the required key sequence for unlocking a door, or augmented notes on a guitar amplifier for different settings with different zoom levels. Their interaction technique allows users to put self-authored instructions as notes onto physical objects. It enables a dynamic and editable visualization for guidance of the operation.

Micro AR [104] added a magnifying glass on a mobile phone's rear camera and presented a hybrid physical/virtual hypermedia that integrates paper and AR using the metaphor of a loupe tool to represent magnification functionality (Figure 2.17). Making use of the familiar action of observing small objects as part of the metaphor, the system embedded intuitive and semantic browsing operations into paper media through a magnified peephole AR interface.

FIGURE 2.17: Augmented Encyclopedia consists of a magnified peephole interface and micro AR markers [104].

In addition to the camera, other embedded sensors on the mobile phone have also been used for mobile AR interaction. For example, Ha and Woo [36] demonstrated a mobile phone-based indirect 3D object translation method that used mechanical sensor information in an AR environment. A user can easily use a mobile device equipped with a 2D touch screen, a 3DOF accelerometer and compass sensors to manipulate 3D virtual objects in 3D space by touching the screen as well as rotating the phone itself (Figure 2.18). This result can be used for mobile phone-based 3D user interfaces for AR applications in normal indoor and outdoor environments without incorporating any special tracking installations.

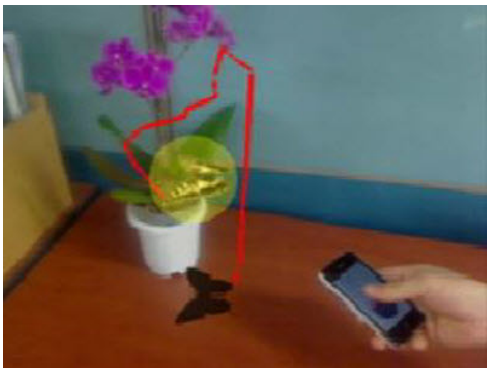

FIGURE 2.18: Moving the virtual butterfly in the space with a sensor-embedded mobile phone [36]

Mossel et al. [80] described two novel intuitive 6DOF manipulation techniques, 3DTouch and HOMER-S for mobile AR applications (Figure 2.19). These were aimed at reducing or even eliminating touch input to avoid abstraction levels. 3DTouch provides 6DOF transformations by separating degree-of-freedom and combining one- and two-finger touch input with the device pose, while HOMER-S completely decouples object transformation from the limited mobile screen space. It does not require any touch input, and uses only the device pose to provide full 6DOF transformations. User studies with both techniques found them to be an intuitive way to translate and rotate virtual objects.

Mossel et al. [81] also present a novel selection technique, DrillSample, which requires only a single touch input for selection and preserves the full original spatial context of the selected objects. This allows disambiguating and selection of strongly occluded objects or of objects with high similarity in visual appearance (Figure 2.19 right). A user study with the system found that DrillSampe achieves significant performance increases in terms of speed and accuracy over more traditional techniques.

Jin and Park [50] enhanced the mobile AR interaction from a novel tangible

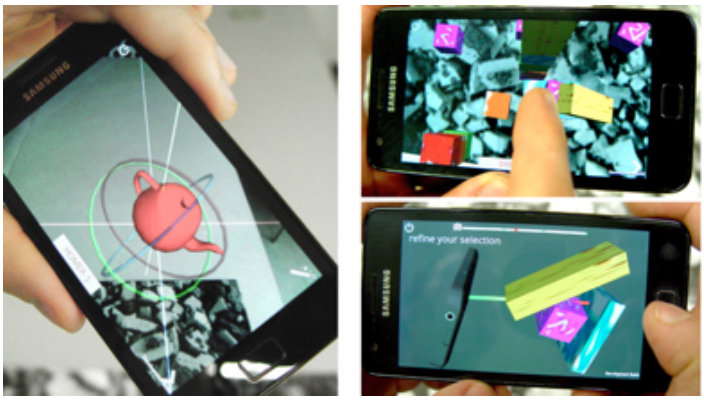

FIGURE 2.19: Using HOMER-S for translation and rotation (left) [81], and selecting a partly occluded object (pink cube) using DrillSample selection technique (right) [80].

perspective. Their system tracks a visual marker on a tactile pad that provides vibro-tactile feedback for a realistic immersive experience. The mobile phone provides a variety of vibration patterns to the human palm to represent the augmented object's movements and location. The intensity of vibration can be generated based on the z-axis coordinates of the augmented object. Through a simple memory test application, it was shown that this system is useful for providing intuitive knowledge of the augmented object's spatial and directional information.

In summary, for device-centric interaction in mobile AR the user performs indirect manipulation of 3D virtual objects by directly changing the state of the mobile phone instead of the virtual content. The device-centric interaction for mobile AR, especially like some Tangible User Interfaces (TUIs), can give the user vibrative and/or tactile feedback to provide a clearer understanding of the operation status, and might work better for certain tasks which require more precise strength or position sensing on a 2D surface, like the AR annotation drawing [47]. However, this type of interfaces confined the user input in the limited physical form and force common users to learn new designed input methods that mainly fit the device's characterizes at the first place. In contrast, most mid-air input tasks like the 3D drawing can be performed much natural with the finger tracking. In earlier research the following types of device-centric interaction have been used:

- Keypad input
- Touch-screen interaction
- Camera motion input
- Sensor-based interaction

2.2.2 User-centric Interaction Methods

To provide full 3D manipulation by touch in an integral way, existing device-centric interaction techniques for mobile AR often use the multi-touch capabilities of the device's display with complex multi-finger and hand gestures. However, these are often difficult or impossible to use for one-handed mobile AR scenarios, and their usage requires prior knowledge. Furthermore, a phone's touch screen offers only two dimensions for interaction and limits manipulation to the physical screen size.

Unlike the device-centric interaction methods that heavily rely on the device as an intermediary, user-centric interaction directly accepts the user's natural behaviours, like free-hand gesture, gaze or speech, as input signals. Considering the common usage of mobile devices such as when a user holds a mobile phone with one hand while the other hand is free, researchers have begun to explore alternatives to touch screen-based interaction, such as mid-air gesture-based interactions using finger or hand tracking.

Early methods for gesture input required the user to have a trackable marker attached on the fingers so the mobile camera could track their position. Using the mobile phone's front camera, Henrysson et al. [40] experimented with both 2D gesture input using motion flow tracking, and 3D gesture input using ARToolKit tracking of a fiducial marker attached on the fingertip to interact with the virtual content (Figure 2.20). They compared gesture input to tangible input, keypad interaction and phone tilting in increasingly complex positioning and rotation tasks in a mobile AR context. Usability experiments found that tangible input techniques are best for translation tasks, while keypad input is best for rotation tasks.

FIGURE 2.20: 3D painting using the fingertip attached with the fiducial AR marker [40].

Hürst and Wezel [48] investigated the potential for finger-based interaction for mobile AR applications by applying a colour marker on the fingertip (Figure 2.21). Compared to touch screen and orientation sensor input, they found that finger-based interaction produced the worst performance, but it had a high score of fun-engagement

level that indicated the potential for gaming and other leisure applications.

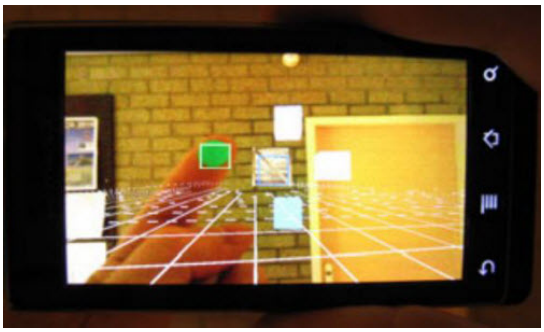

FIGURE 2.21: Using a green marker for fingertip tracking to locate the AR cube [48].

Hürst and Wezel also extended the single-color tracker to two markers: a green and red one on the user's thumb and index finger, respectively [49]. They conducted an evaluation of two finger interaction in a mobile AR board game (Figure 2.22), with translation, scaling, and rotation of two different virtual objects. The user study results confirmed that this nicely integrated virtual and real world interactions in a natural and intuitive way. In addition, it also showed that using more than two markers significantly reduces tracking performance and becomes hard to handle because of finger occlusion.

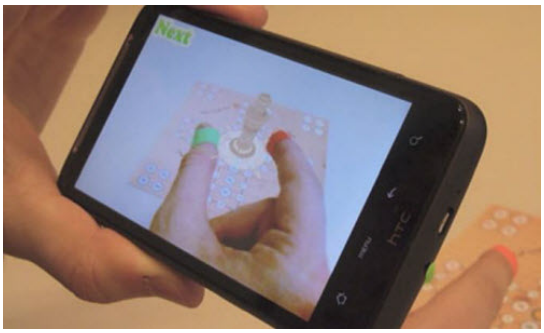

FIGURE 2.22: A green marker (thumb) and a red one (index finger) are used to track fingers for manipulating virtual objects in an AR board game [49].

Attempting to remove the dependence of the marker, Baldauf et al. [13] developed a visual markerless fingertip detection engine that can be used to manipulate virtual 3D objects in mobile AR environments. This can help a user to select one virtual object out of several displayed by pointing at it, or grabbing, dragging and dropping it with a similar gesture or a pinch gesture with thumb and index finger (Figure 2.23). In the case of animated virtual characters in mobile AR games, for example, the detection of the user's fingertip allows for gestural interaction with the character which may react to these gestures in different ways, e.g. by following a fingertip.

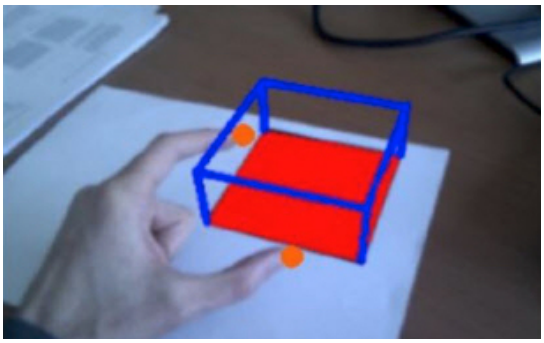

FIGURE 2.23: Using the pinch gesture to change the size of the AR cube [13].

Seo et al. [99], proposed a new method that made the use of the user's whole hand for mobile AR interaction. With the help of a palm pose estimation method that is unaffected by finger motions, the application can support natural augmentation anytime and anywhere, so a user can freely interact with AR content without using visual markers or tags. For instance, a virtual object can be placed on the palm, and then moved along with it (Figure 2.24), and the user can interact with the virtual object by opening or closing motions of the hand. This work was further developed by Choi et al. [23], in which the application provides tactile interactions with the virtual object through a tactile glove with vibration sensors, so the user can perceive realistic sensations from the virtual object in both visual and tactile ways.

FIGURE 2.24: Using the palm to play with the AR pet on a mobile phone [13].

These methods rely solely on skin colour to segment the hand, making it difficult to continuously recognize the hand in an environment with skin-coloured materials. Chun and Höllerer [24] solved the skin colour issue by using a textured target as the background to subtract the hand region and detect the fingers, and then recognized a pinch motion for translation and scaling. However, experimental results indicate that this type of hand interaction might not be a good fit for subtle hand motions.

Compared with the mobile AR interfaces on the smart phone, gesture interaction

techniques are preferred for mobile AR applications running on lightweight wearable AR systems. This is because they can provide natural interaction with less disruption and distraction for the user. The user can still focus their attention on the screen information and the task itself while selecting menus with hand input, without trying to separately locate the touch pad and understand the input mapping. This could potentially reduce errors and improve safety issues.

There have been a few earlier systems that explored gesture input with wearable computing. One of the first was Tinmith [88] that used special gloves that allowed a user to select menu options and specify 3D input in an outdoor AR interface (Figure 2.25). However, in Tinmith only a single point on each hand was tracked by computer vision, providing only coarse gesture input.

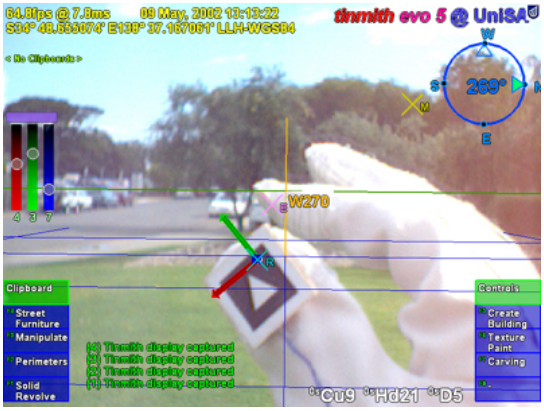

FIGURE 2.25: Modelling cursors with pinch gloves for Tinmith [88]

Other systems such as Kurata's hand mouse [63] provided 2D input in wearable systems, while Kolsch's multimodal interface [62] and HandVu [61] performed 3D hand tracking based on skin segmentation (Figure 2.26). Users could select menu objects or point to interact with objects at a distance, but there was no creation of a full 3D hand model for richer interactions. Starner et al. [103] demonstrated how vision based hand tracking can be used for sign language recognition on wearables, while Rekimoto [91] used a wrist mounted camera to track fingertips for command input. Moeslund [79] and Kölsch [60] provide summaries of these and other wearable gesture based systems.

Gaze information has also been adopted in wearable AR to perform actions in concurrence with gesture interaction. Höllerer suggested that gaze interaction is an effective interaction technique in optical see-through HMD based wearable AR [44]. Gaze interaction has been considered to be an easy, natural, and fast way of interacting in virtual environments in spite of the so-called "Midas Touch" problem in which every object being looked at is selected [106]. As a solution for this problem,

FIGURE 2.26: Controlling mouse and button interface with hand gestures (left); Interacting with virtual objects and ARToolKit (right) [61].

Tanriverdi has suggested a histogram that represents the accumulation of eye fixations on each possible target object [106]. The largest histogram values mean that the object has the highest interest of the user, and the latest object with the highest interest would be selected in this system.

Similarly, Park et al. proposed an "Aging" technique [87] by selecting the "oldest" object with the most accumulated interests instead of the "recent interest" object (Figure 2.27). In this case, each object has its own age that represents a degree of interest. The object gazed at continuously by a user is getting older while the object not gazed at is getting younger. With Park's gaze interface, the user's selection would be settled as long as the object is the oldest, providing satisfying mobility and convenience of viewing in an AR gallery scene. This is different from earlier systems in which if the user turn his/her eye one from one object to another, the selection would be changed.

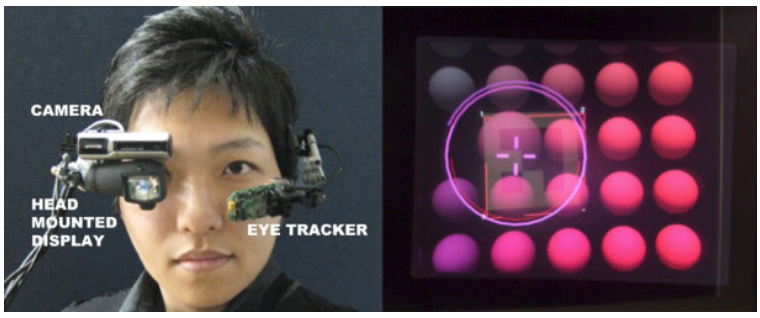

FIGURE 2.27: Wearable Augmented Reality System - HMD with eye tracker (left); A user selects a sphere viewed through a calibrated optical see -through display by gazing (right) [87].

In summary, user-centric interaction in mobile AR involves natural input from the user without any direct input on the mobile device, allowing the user to directly control virtual objects by manipulating them in the real world. Gesture-based interaction techniques in mobile AR mainly include:

- Marker-based fingertip input

- Markerless fingertip input
- Markerless full-hand input

2.3 Free-hand Gesture-based Interaction

Even though the research work described in Section 2.2.2 showed the potential of finger-based or palm-based gesture input using computer vision techniques for mobile AR applications, one shortcoming was the lack of accurate depth sensing data that can be used by mobile phones to understand markerless 3D gesture inputs. Full hand pose recognition can support a rich spatial interaction experience, but typically requires customized hardware like Digits [55], a complicated setup like 6D Hands [116], or expensive computational resources like Handpose [100], which are currently not feasible or available for mobile devices. So fully-fledged 3D gesture interaction for AR on mobile devices has not been well investigated.

Recently researchers have begun prototyping to explore the use of depth cameras for input on wearable computers and mobile devices to support more complicated spatial hand input in AR environment. For example, Harrison et al. presented Omni-Touch [37], a wearable depth-sensing and projection system that enables interactive multi-touch applications on everyday surfaces without other instrumentation of the user or environment except a shoulder-worn system, as shown in Figure 2.28. The wearer can use their hands, arms and legs as graphical, interactive surfaces, and can also appropriate surfaces from the environment to expand the interactive area (e.g., books, walls, tables). On such surfaces, OmniTouch provides capabilities similar to that of a mouse or touchscreen without any extra calibration, enabling a wide variety of on-the-go interactions.

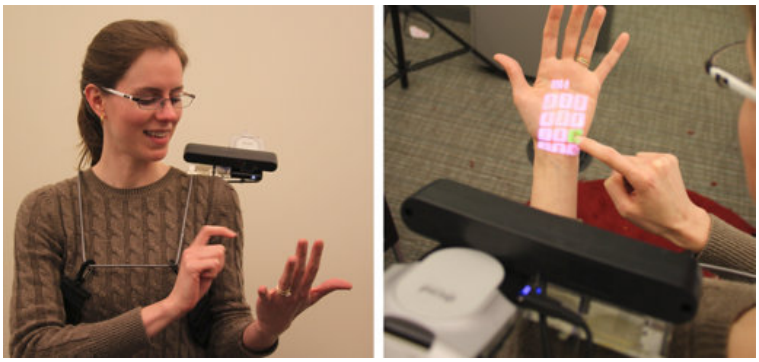

FIGURE 2.28: OmniTouch is mounted on the user's shoulder, and allows everyday surfaces to be appropriated for graphical multi-touch interaction via the depth sensing and projection [37].

Benko et al. demonstrated MirageTable [15], which is instrumented with a single depth camera, a stereoscopic projector, and a curved screen, and designed to merge real and virtual worlds into a single spatially registered experience on top of a table. The depth camera tracks the user's eyes and performs a real-time capture of both the shape and the appearance of any object placed in front of the camera (including user's body and hands). This real-time capture enables perspective stereoscopic 3D visualizations to a single user that account for deformations caused by physical objects on the table. In addition, the user can interact with virtual objects through physically-realistic freehand actions without any gloves or trackers.

Wang et al. developed 6D Hands [116], a bimanual hand tracking system that provides physically-motivated 6DOF control for 3D assembly. The system uses two consumer-grade webcams to observe the user's hands, and solves the pose estimation problem with efficient queries of a precomputed database that relates hand silhouettes to their 3D configuration (Figure 2.29). A major feature of their system is that it tracks the hands without gloves or markers, leaving them unencumbered to use the keyboard and mouse. By facilitating these mixed-mode operations, the system serves as a practical complement to the mouse and keyboard for 3D assembly interactions.

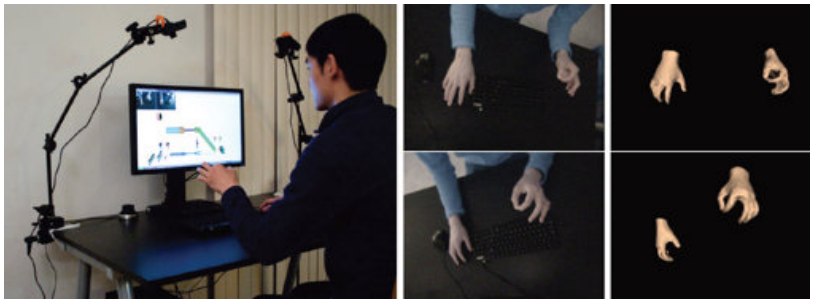

FIGURE 2.29: System setup with two RGB-Depth cameras (left); Pose estimation generated based on the camera view (right) [116].

Ha et al. introduced [35], which attached two depth cameras to the front of a head mounted display to support full hand gesture interaction in a wearable AR environment. WeARHand uses no environmentally tethered tracking devices and localizes a pair of near-range and far-range RGB-Depth cameras mounted on a head-worn display and a moving bare hand in 3D space by exploiting depth input data (Figure 2.30). The near-range camera is used to detect the hand and close-up occlusion, and depth perception is enhanced through egocentric visual feedback, including a semi-transparent proxy hand. The long-range camera is used to support occlusion and shadowing of more distant objects and to determine the correct scale relationship between the physical and virtual worlds. Thus, the camera pair can cover the combined range (0.15m – 3.5m) needed to track both the hands and the

environment. The proposed hand-based user interface can be applied to many other 3D interaction scenarios in wearable AR environments, like art, design or exposure psychotherapy.

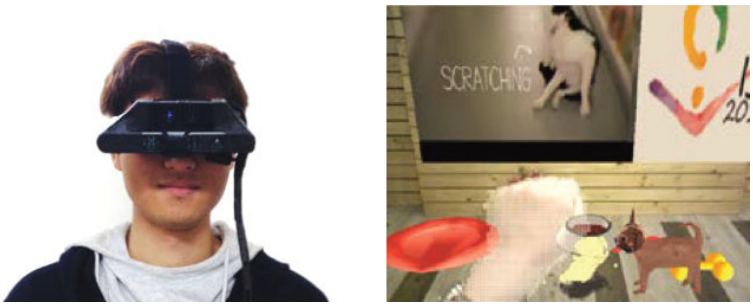

FIGURE 2.30: The user wears head-worn display with near and far-range RGB-D cameras (left); The user can manipulate virtual 3D objects with proper depth perception through visually enhanced feedback (right) [35].

Kim et al. presented a wrist-worn system for sensing the full 3D pose of the user's hand [55], enabling a variety of freehand interactions on the move. Its electronics are self-contained on the user's wrist, but optically image the entirety of the user's hand (Figure 2.31). This data is processed using a new pipeline that robustly samples key parts of the hand, such as the tips and lower regions of each finger. These sparse samples are fed into new kinematic models that leverage the biomechanical constraints of the hand to recover the 3D pose of the user's hand. The system works without the need for full instrumentation of the hand (for example using data gloves), additional sensors in the environment, or depth cameras which are currently prohibitive for mobile scenarios due to power and form-factor considerations. Digits targets mobile settings, and is specifically designed to be low-power and easily reproducible using only off-the-shelf hardware that is both smaller and more power efficient, and can be utilized for 3D spatial interaction with mobile devices, and eyes-free interaction on-the-move, and gaming.

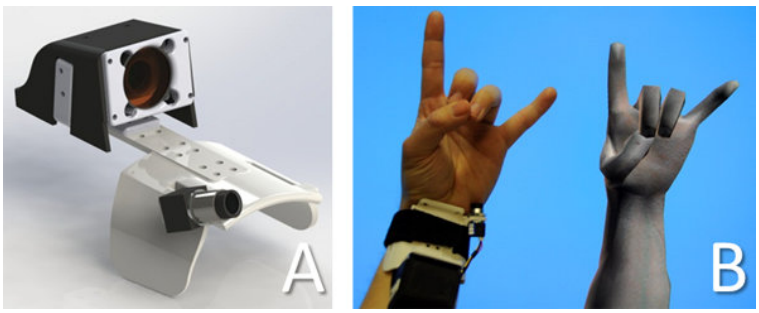

FIGURE 2.31: Digits is a wrist-worn sensor for freehand 3D interactions on the move (left); Digits recovers the full 3D pose of a user's hand (right) [55].

Fully articulated hand tracking promises to enable fundamentally new natural and intuitive interactions with virtual and augmented worlds.

2.4 Conclusion

In this chapter we have reviewed research that demonstrated some successful examples of how 3D gesture interaction could be supported in AR environment with desktop computing resources, however they are not yet adopted on mobile devices.

Our research explores the usability of free-hand gesture-based interaction on the current generation of smartphones with and without built-in RGB-Depth cameras in mobile AR scenarios. This is an area which has not been well explored in previous research. We will evaluate the usability of our proposed systems, and also summarize the general design principles of mobile AR interfaces through user studies, which could be applied as guidance for the development of mobile AR applications.

Chapter 3

Free-hand Fingertip-based 2D & 3D Interaction

A variety of interaction techniques have been used for mobile AR systems to manipulate augmented content. Most of these techniques often focus on device-centric methods based around touch input. For example, people scroll their fingers on the touch screen to translate a 3D virtual cube, or rotate their mobile phone with the built-in gyroscope to spin a virtual cylinder. However, the touch abstraction layer still exists, and the mapping between the touch input and virtual object manipulation requires conversion from 2D surface to 3D space which is often confusing and limited to the screen size of the mobile device. More importantly, users may not be able to easily interact with virtual objects if they are holding the mobile device with one hand and touching the screen with the other, while at the same time trying to maintain visual tracking of an AR marker.

In this chapter, we report on two gesture-based interaction methods for handheld mobile AR systems that overcome these problem, and offer interesting alternatives to touch-screen based interaction methods for virtual object manipulation. In Section 3.1 we first present a 2D markerless fingertip-based gesture input technique, including the development of our own mobile AR tracking system (Section 3.1.1), and study its usability in Section 3.2 by comparing it with a freeze view touch method (Section 3.2.1) and a common free view touch-based interface for virtual object manipulation. The 2D gesture interaction method is able to overcome the screen shaking that often occurs when using current touch-screen input techniques, but there is a lack of accurate depth sensing of fingertips. Since the AR content is overlaid on a view of the physical world, 3D user interfaces can be used more naturally and efficiently for 3D manipulations instead of conventional 2D interaction techniques. In Section 3.3, we then describe a technique for 3D markerless fingertip-based interaction using an RGB-Depth camera attached to the handheld device, and in Section 3.4 conducted a

user study to evaluate the method for different AR scenarios.

3.1 2D Markerless Fingertip-based Interaction

3.1.1 Feature Tracking on a Mobile Phone

As we discussed in Section 2.1.3, visual tracking is a key enabling technology for AR systems, and modern AR applications prefer NFT methods instead of the marker-based tracking because of more robust performance. However, most feature generation has huge computational complexity, which demands high computational power. It is therefore difficult to use in mobile AR applications, and the most critical issue is how to maintain sufficient accuracy and robustness at the same time.

In this section, we optimize an algorithm based on modified state-of-the-art feature descriptors for mobile devices, and present a real-time tracking system, consisting of keypoint detection, description and matching, for textured planar targets on current-generation mobile phones.

Keypoint Detection and Description

Effective and efficient generation of keypoints from an image is a well-studied problem in the research literature and forms the basis of numerous computer vision applications. Decomposing an image into local regions of interest or features is a widely applied technique in computer vision and used to alleviate complexity while exploiting local appearance properties. Image representation, object recognition and matching, 3D scene reconstruction and motion tracking all rely on the presence of stable, representative features in the image.

The ideal keypoint detector finds salient image regions such that they are repeatedly detected despite change of viewpoint; more generally it is robust to all possible image transformations. Similarly, the ideal keypoint descriptor captures the most important and distinctive information content enclosed in the detected salient regions, such that the same structure can be recognised if encountered.

Amongst the highest precision features currently in the literature is SIFT [75], which combines a scale and orientation invariant region detector and a descriptor represented by a 3D histogram of gradient distribution in the detected regions. The SIFT descriptor has high descriptive power and robustness to illumination and viewpoint changes. However, because of high dimensionality of its description vector, it is too computationally intensive and prohibitively slow for real-time applications with any complexity image.

The SIFT descriptor has been extended in several ways, for example, PCA-SIFT [54] decreased the descriptor dimensionality by using Principal Component Analysis (PCA) which normalizes gradient patches instead of histograms. However, it is more compact than the standard SIFT descriptor, compromising its distinctiveness and increasing the time for descriptor formation which almost removes the increased speed of matching. GLOH [78] was another extension of the SIFT descriptor designed to increase its robustness by using a polar grid instead of a regular Cartesian coordinate system. It had been shown to be more distinctive but also more expensive to compute than SIFT. Two modifications of the GLOH descriptor, sGLOH and sGLOH+ [14], achieved rotation invariance without rotating the feature patch before computing the descriptor, since predefined discrete orientations can be easily derived by shifting the descriptor vector. A test of these descriptors produced good results in terms of robustness and stability, at a reasonable increase in the computational cost.

On the other end of the spectrum, a combination of the FAST keypoint detector [95] and the BRIEF [19] approach to description offered a much more suitable alternative for real-time applications. BRIEF was designed for super-fast description and matching and consists of a binary string containing the results of simple image intensity comparisons at random pre-determined pixel locations. Despite the simplicity and efficiency of this approach, the method is very sensitive to image rotation and scale changes, restricting its application to general tasks. ORB [96] attempted to overcome BRIEF's limitations by applying oriented FAST detector and rotated BRIEF features. It is rotation invariant and resistant to noise, but one issue that has not been adequately addressed yet is scale invariance.

The growing demand for high-quality, high-speed features for mobile AR tracking has led to more research towards algorithms which can process richer data at higher rates. Compared with previously significant algorithms, BRISK [73] is one of the most appropriate algorithms for mobile AR tracking, achieving an excellent balance of accuracy, robustness and speed. The key to its speed lies in the application of a novel scale-space AGAST detector [76] in combination with the assembly of a bit-string descriptor from intensity comparisons retrieved by dedicated circular sampling of each keypoint neighbourhood. The unique properties of BRISK can be useful for a wide spectrum of mobile AR applications, in particular for tasks with hard real-time constraints or limited computation power: BRISK ultimately offers the quality of high-end features in such time-demanding applications.

Binary Robust Invariant Scalable Keypoints (BRISK)

The BRISK is composed of three main steps: keypoint localization, feature description and feature matching.

1. Scale-space Keypoint Detection

Points of interest are identified across both the image and scale dimensions using a saliency criterion. To boost efficiency of computation, keypoints are detected in octave layers of the image pyramid as well as in layers in-between. The location and the scale of each keypoint is obtained in the continuous domain via quadratic function fitting, as shown in Figure 3.1.

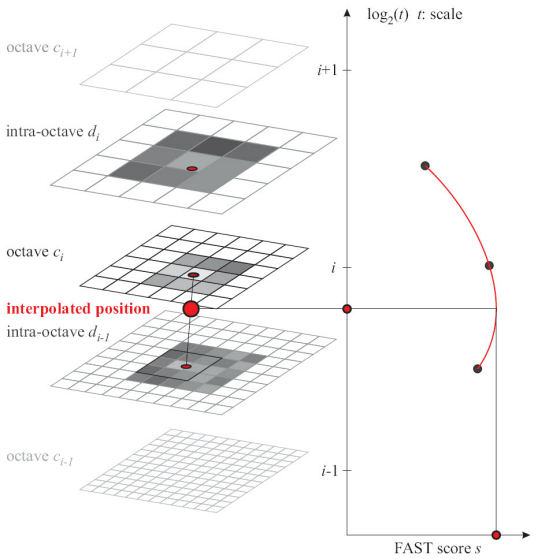

FIGURE 3.1: Scale-space interest point detection: a keypoint (*i.e.* saliency maximum) is identified at octave c_i by analyzing the 8 neighboring saliency scores in c_i as well as in the corresponding scores-patches in the immediately-neighboring layers above and below [73].

2. Keypoint Description

A sampling pattern consisting of points lying on appropriately scaled concentric circles (Figure 3.2) is applied at the neighbourhood of each keypoint to retrieve grey values: processing local intensity gradients, the feature characteristic direction is determined. Finally, the oriented BRISK sampling pattern is used to obtain pairwise brightness comparison results which are assembled into the binary BRISK descriptor.

3. Descriptor Matching

Matching two BRISK descriptors is a simple computation of their Hamming distance as done in BRIEF: the number of bits different in the two descriptors is a measure of their dissimilarity. Notice that the respective operations reduce to a bitwise XOR followed by a bit count, which can both be computed very efficiently on today's processor architectures.

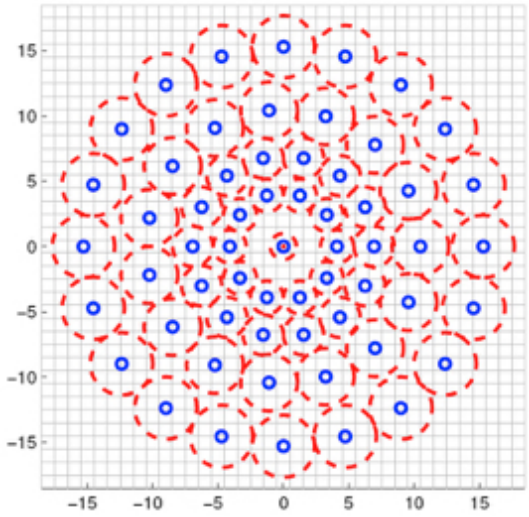

FIGURE 3.2: The BRISK sampling pattern with N = 60 points: the small blue circles denote the sampling locations; the bigger, red dashed circles are drawn at a radius σ corresponding to the standard deviation of the Gaussian kernel used to smooth the intensity values at the sampling points [73].

Make BRISK Feasible on a Mobile Phone

In our research we chose BRISK as the core algorithm for the NFT implementation of our mobile AR system, and so needed to port the detector to a mobile platform. It should be noticed that the original source code of BRISK is only for traditional desktop systems and optimised for the Intel CPUs with specific assembly language, so it is not suitable for mobile platforms. Thus, we started porting BRISK to Android phones and implementing a mobile AR system with extra optimisations. We tested a Samsung Galaxy S2 (ARM 1.2 GHz Dual-Core Chipset, 1GB RAM, Android 2.3 OS) as the target mobile platform, and ported the BRISK code in C++ using the Android NDK[1]. The computer vision part was developed partly using OpenCV4Android[2], and the time-consuming steps of BRISK was rewritten using the FastCV library[3], which enabled the NEON[4] acceleration for ARM chips.

Finally, we combined the detection and description parts with optical-flow tracking to create a fully-featured robust AR tracking system. Figure 3.3 illustrates the flow model outlining our algorithm structure. Each frame captured from the mobile camera was checked by the AGAST detector to detect interesting keypoints, and then the BRISK descriptor generates the description of each keypoint. Feature matching with a pre-defined target was conducted and some low-quality points were removed

[1] http://developer.android.com/ndk/index.html
[2] http://opencv.org/platforms/android.html
[3] http://developer.qualcomm.com/software/fastcv-sdk/
[4] http://www.arm.com/products/processors/technologies/neon.php

to reduce the calculation noise. Successfully matched points were used to calculate the camera pose that was refined later to make the result more accurate and stable. At this moment, the system automatically starts to track the matched points instead of processing feature extraction, which can offer a huge calculation saving and a faster running performance.

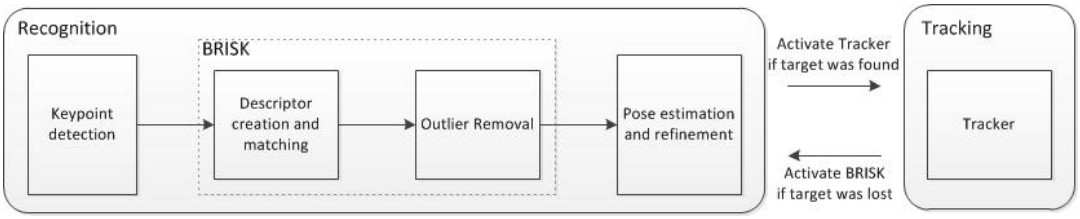

FIGURE 3.3: The work flow model of BRISK-based NFT algorithm.

The goal for our mobile AR system was to achieve real-time tracking performance of at least 15 Frames Per Second (FPS) (a conservative minimum for interactive systems), and to strive for 30 FPS, which will provide enough room for the reduction in frame rate caused by the further hand gesture detection. Table 3.1 lists the speed of the camera pose estimation procedures achieved in the ported software.

TABLE 3.1: Processing time for 800 by 480 resolution

Procedure	**Processing Time (msec)**	
RGB2GRAY	6.04	
Keypoint detector	25.95	(83 keypoints detected)
Descriptor extractor	12.19	
Descriptor Matcher	16.80	(30 keypoints matched)
Homography	2.59	
Total	63.57	

The feature detection, description and matching have already been implemented as shown in the Figure 3.4. Our tracking software can run smoothly and robustly on the target phone at an interactive frame rate of around 15 to 20 FPS with 800 by 480 pixel camera resolution.

We also completed a Unity3D plug-in[5] for the tracking part to load 3D models into the tracked AR scene (Figure 3.5). This enables us to use the Unity3D game engine to more conveniently develop mobile AR applications.

[5] http://www.unity.com/

FIGURE 3.4: BRISK feature detector, descriptor, and matching running on an Android smartphone.

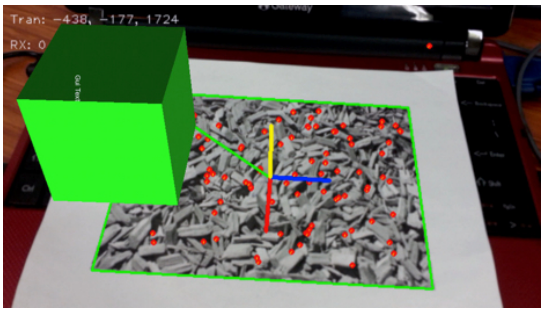

FIGURE 3.5: 3D cube is loaded into the tracking scene with our tracking plugin in the Unity3D game engine.

3.1.2 Fingertip Detection on a Mobile Phone

In this section, we describe an improved algorithm to complete hand segmentation and fingertip detection for mobile devices.

At the early stage, we aimed at methods using an explicitly defined skin region as they can provide real-time hand detection, especially for handheld devices. The fast and easily way to segment a skin colour area from one image is to build a skin classifier based on explicitly defined boundaries in some colour space through a set of rules. The non-parametric skin modeling method detects the skin colour area by using training data which has been calculated and prepared before the system starts, such as a lookup table [71]. For each pixel, it only needs to check its skin colour probability from the lookup table; therefore, it is considered to be fast in

processing. However, it requires a large storage space for the training data, which is a challenge for most mobile devices. Furthermore, the performance of this method directly depends on the training images collected. If the set of the training data is too small, the performance may be even worse than the method based on explicitly defined boundaries.

In order to improve the accuracy of the training data in non-parametric skin modeling, a new parametric skin modeling method, which uses a Gaussian function to calculate the skin colour distribution, has been developed, trying to segment the hand region anywhere (against any background environment) and anytime (in any normal light condition). For example, the Handy AR [71] segmentation procedure used training data set and a lookup table to detect the hand region. This method also can perform in real-time and work more robustly in different background environments and lighting conditions. However, some changes and improvements need to be carried out for our mobile AR system.

We combined the Ostu threshold algorithm [86] and the general skin colour model to detect skin colour pixels, which used an adaptively learned histogram together with a lookup table to refine the hand region. Our approach works as follow. The input image is first converted into a grey scale image. The Ostu algorithm and a lookup table trained from a set of collected data are used to separately segment the skin colour region. The outputs of these two methods are compared and only the regions considered to be a skin colour region for both methods are treated as the hand area. Then, a distance transformation will be applied to find the point exhibiting the maximum distance value. The region that includes this point is the hand region and other regions are considered to be noise. Furthermore, this point will be recorded and used for the next frame to avoid mistakes when the hand moves out of the camera view.

To locate the fingertips from the hand region, we first used a curvature-based algorithm from Handy AR. We assume that when the curvature of the hand contour is larger than a defined threshold (here we define the curvature as the angle cosine value and the threshold is 0.8), it is considered as the fingertip. However, as with the filtrated hand region shown in Figure 3.6, the hand region may be cut by the screen boundary. The two intersection points always have large curvature values and may be mistakenly considered as the fingertips by the algorithm.

To solve this issue and locate the real fingertips, we found an alternative fingertip tracking approach based on Rajesh's method [90]. As we described, when we separated the hand contour in one image, we also locate the point that has the maximum

FIGURE 3.6: Filtrated hand region (left); Finger region (middle); Fingertips located (the blue point is the one has the maximum distance value in distance transformation) (right).

distance value in distance transformation and consider it as the centre point of the palm. In addition, the maximum distance D is considered half of the palm width and twice of one finger width. Then the fingers can be eroded completely by using an element with the width D/2 to only leave the palm region; therefore, the finger areas can be calculated by erasing the palm region from the whole hand region. For each finger area, we find the minimum area rectangle (a rotated rectangle fitted to the finger area) and calculate the midpoints of the four edges of the rectangle. The fingertip position is defined as the midpoint that is farthest from the centre point of the hand region, as shown in Figure 3.6. The final fingertip detection ran around 20 FPS with 800 by 480 pixel camera resolution on our test smartphone with the built-in camera (Samsung Galaxy S2).

3.1.3 Interaction Design

In this section we describe our gesture-based interface design based on developed camera and hand tracking technologies described in previous two sections. In mobile AR applications, a user usually holds a mobile phone with one hand while using the other hand for interaction. Thus, researchers have mainly studied one handed gesture-based interactions using finger detection or hand tracking. Early methods required the user to have a trackable marker on the fingers, but the latest markerless gesture interaction helps users to interact with mobile AR applications using their bare hands. The camera on the back of the mobile device can track the user's fingertips and use that to directly interact with the virtual content in the AR scene. The tracked fingertips can be treated as key input points like the cursor on the desktop computer. In our research we only consider the condition where the hand is placed in front of the user's face, and when at least one or two fingers are visible during the gesture interaction. This assumption is valid in most handheld mobile AR situations.

As reported in literature by Martinet [77] and Veit [109], separating the DOF of an interaction task leads to better results than trying to use the separated DOF of a multi-touch display in an integral way, as demonstrated in related work [80]. In out interface we perform mode changes and object selection by using the touch input, and then use gesture input for performing translation, rotation and scaling separately to alter the location, pose and size of virtual objects in the AR space, respectively.

To perform mode changing and object selection through touch input, the user performs a long press on the touch screen once the AR tracking is established, and a context menu appears around the pressed point (Figure 3.7), implemented as a Marking menu [64]. The menu has three buttons: trans (translation), rot (rotation) and scl (scaling). We choose this menu style to reduce the user's finger movement and performance time. The user can only have one menu button selected at a time, and once pressing the button the system will switch into a specific operation mode in which only the selected manipulation can be performed on the virtual object. For example, once they have pressed the "*scl*" button the gesture input will scale the selected virtual object.

The world coordinates appear at the centre of the virtual object to indicate the virtual object's position and pose (Figure 3.7). The user can then choose which axis he/she wants to operate along by clicking the button at the end of the axis. There will be no influence on the other axis, and multiple axes cannot be selected at one time. If the user wants to change the manipulation mode or axis, they can perform another long press on the screen to call out the main menu and repeat the previous steps mentioned above.

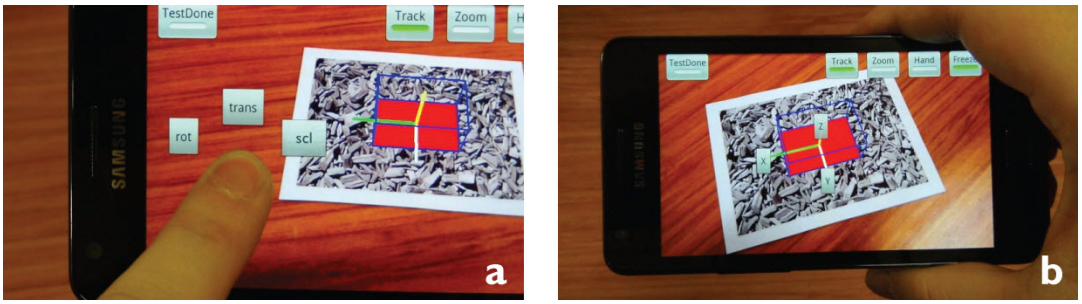

FIGURE 3.7: The Marking menu (left) and coordinate buttons (right).

Once the mode and object are confirmed, the user can click on the "*Hand*" button on the screen to enable the natural gesture interaction mode. From here on, if the detected hand area is bigger than a threshold value, the hand region inside the camera image will be segmented from the background. The most prominent fingertip (main fingertip) will be identified and marked by a small white circle in the Translation and

Rotation modes, while a second fingertip (secondary fingertip) will also be recognized and marked by a small red circle in the Scaling mode.

We switch between activated and deactivated states for the main fingertip using a traditional dwell time selection technique. If the user keeps the chief fingertip relatively still in a tiny region (10 by 10 mm in our case) around the current position longer than a certain time (2 seconds in our case), the state of the chief fingertip will be reversed, causing the appearance or disappearance of a big white circle (Figure 3.8). The big white circle will be initially drawn when the interaction starts to highlight the main input fingertip, and is also to visually indicate whether the real-time gesture input is activated or not for the user. For example, if the big white circle appears, it means that all the gesture inputs will be deactivated and ignored, and will have no practical influence on the virtual object's state to avoid potential unintentional input by mistake. On the other hand, when the big white circle disappears, it indicates that only now the user can make actual operations on the virtual objects with their finger gestures. This approach is based on the idea that there is a virtual tangible surface in the air, and the big circle demonstrates whether the fingertip is touching this "*surface*" or not. While waiting for the locking state change of the fingertip, there is a small green line growing from the centre to the edge of the small circle to visually indicate the waiting process.

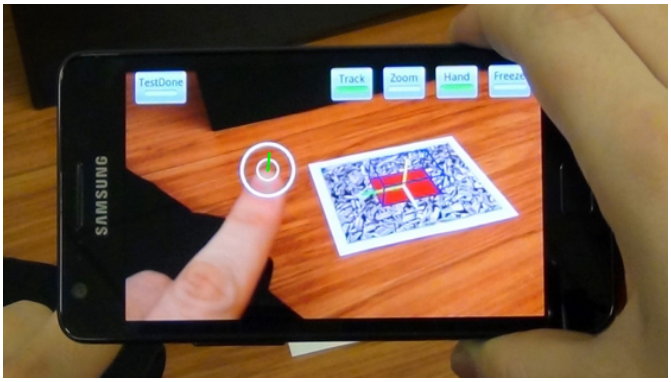

FIGURE 3.8: The dominating fingertip is circled by a small white circle, and the dwell time countdown is indicated by a green line from the fingertip. A big white circle shows that the interaction is locked and deactivated.

The position change of detected fingertips in the camera image coordinate system will be converted into a change of the virtual object's position, pose or scale. Specifically, we map the position change of a single fingertip moving in the air onto the object translation and rotation values, while the object scale value is controlled by the distance change between the two fingertips as they move in a pinching motion.

3.1.4 Prototype Implementation

Our mobile AR system consists of the NFT tracking module made in Section 3.1.1 to identify an image and the markerless fingertip detection module developed in Section 3.1.2 to convert gesture commands to augmented object rendering parameters. The two parts share the state variables of the virtual object during the application life-cycle. In addition, the system offers a basic User Interface (UI) for human-computer interaction. The touch interaction and UI layout are built in Java with the Android SDK[6], which provides convenient and powerful UI functions for touch gestures.

As we mentioned before, the tracking software can run smoothly and robustly on the Samsung Galaxy S2 at an interactive frame rate of around 15 to 20 FPS with 800 by 480 display pixel resolution. Combining the fingertip detection part into the AR prototype system on the phone, we achieved an interactive performance of 10 to 15 FPS, which is sufficient for manipulating virtual objects, as shown in Figure 3.9).

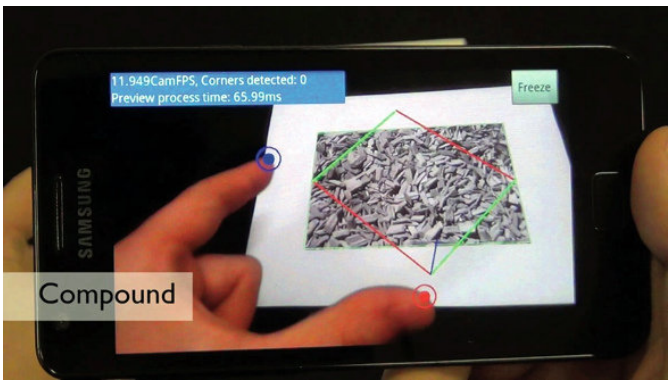

FIGURE 3.9: Using fingers to rotate and change the size of the rectangle at the same time.

3.2 User Study

To investigate the usability of our 2D fingertip-based gesture input method, we also developed a freeze view touch method for mobile AR applications, as described in Section 3.2.1, and then conducted a formal user study comparing these two methods with each other, and with a traditional free view touch approach.

3.2.1 Freeze View Touch Interaction

The freeze view touch based interaction method simulates a fixed view position by freezing the current camera frame and allowing the user to move the device without

[6] http://developer.android.com/studio/index.html

updating the visual tracking. The user can use the touch screen to interact with virtual content while the camera view is frozen. After successful completion of the desired input task, the user can unfreeze the view to restart the tracking thread. This overcomes the problem of the user trying to interact with a virtual object on a live camera view that may be moving as their hand holding the handheld display is shaking.

Researchers have previously explored freezing the camera frame to stabilize the touch screen input while using camera tracking. For example, Langlotz et al. [68] developed a mobile in-situ content augmentation system using stylus input and camera tracking. Manipulation and authoring of the virtual content was done in a view-freeze mode that allowed users to manipulate the virtual content on a still camera image. Guven et al. [34] presented a set of mobile AR interaction techniques to embed multimedia content in the physical environment. Their technique relied on freezing the frame for later editing, which allowed users to capture a snapshot (frame) of the environment to work on, and later map the results back into the physical world. Lee et al. [70] investigated the benefits of using the freezing method in annotation tasks through a user study. However, none of these papers compare the use of input on a frozen view to gesture-based interaction.

To begin the mode and object selection, the user uses the same interaction procedure described in Section 3.1.3. They make a long press on the touch screen, select an input mode from three options, and then choose one axis to start the real-time interaction. Once a trackable target is identified and tracked correctly, the user keeps the mobile phone in an appropriate position with a clear view of the virtual object they want to interact with, and then click on the "*Freeze*" button to freeze the AR view. The freeze button will not be activated if the visual tracking is not working. Once the freeze button has been clicked, the system will immediately stop the tracking thread to save computing resources, and users no longer need to point the mobile phone at the target to see the virtual scene.

The freeze view touch method has the same input mapping design of translation, rotation and scaling input as the gesture interaction, but is based on the touch position on the mobile screen instead of the fingertip position in the mid-air. After completing all the manipulations, the user can click the "*Freeze*" button again to unfreeze the AR scene, remapping the edited virtual objects onto the live tracking image. The main benefit of this approach is that it supports precise editing of virtual content. The traditional free view touch interaction shared most interaction design and operation logic except the freeze view function.

We implemented both the freeze view touch interaction and the free view touch one with the Android SDK, and directly added them on the top of our NFT solution on the same target mobile platform mentioned in Section 3.1.4.

3.2.2 Experiment Setup and Procedure

To evaluate the interaction methods, we set up a within-group user study with one independent variable: the type of interaction method (traditional view touch, freeze view touch, and 2D fingertip-based gesture interaction). Since the skin colour detection of gesture interaction can be easily affected by lighting changes in the environment, our evaluation performance was tested under stable light conditions.

Ten right-handed participants at the university were recruited for the experiment (6 males and 4 females, 6 users at ages 21-30, 3 at ages 31-35, 1 at ages 41-45) (M = 28.7, SD = 5.7). All of them had frequently used touch screen devices, and 6 of them had some experience of using three-dimensional interfaces. However, they all had no previous experience with using handheld AR interfaces. During the experiment tests, eight participants held the device in their right hand and used the left hand for interaction, while for the other two it was the other way around. All of them used the index finger for input. No significant differences could be observed regarding handedness.

Each participant was instructed to perform several experimental tasks using the three interaction interfaces respectively. Interfaces were presented in a different order to the participants to exclude potential learning effects. Three types of manipulations (translation, rotation, and scaling) were included in tasks for each interface. The experimental task was to manipulate a virtual cube in a mobile AR application and match it to the indicated target pose and position. For all tasks, both the indicated target and manipulated cube were clearly displayed on the same screen (Figure 3.10). Subjects could decide themselves when to start the task by clicking the "*Test Start*" button and when to end the task by clicking the "*Test Done*" button. The timing of the task was started after the mode and axis selection were done.

In the translation task, subjects had to move a square on the target XY plane to four target positions in different directions over various distances (Figure 3.10a). In the rotation task, subjects had to rotate the square around the Z-axis clockwise at angles of 25, 120, 225, 330 degrees between the target and initial square pose (Figure 3.10b). In the scaling task, subjects had to change the size of the square to 0.4, 0.8 times smaller and 1.6, 2.3 times larger at the same proportion (Figure 3.10c).

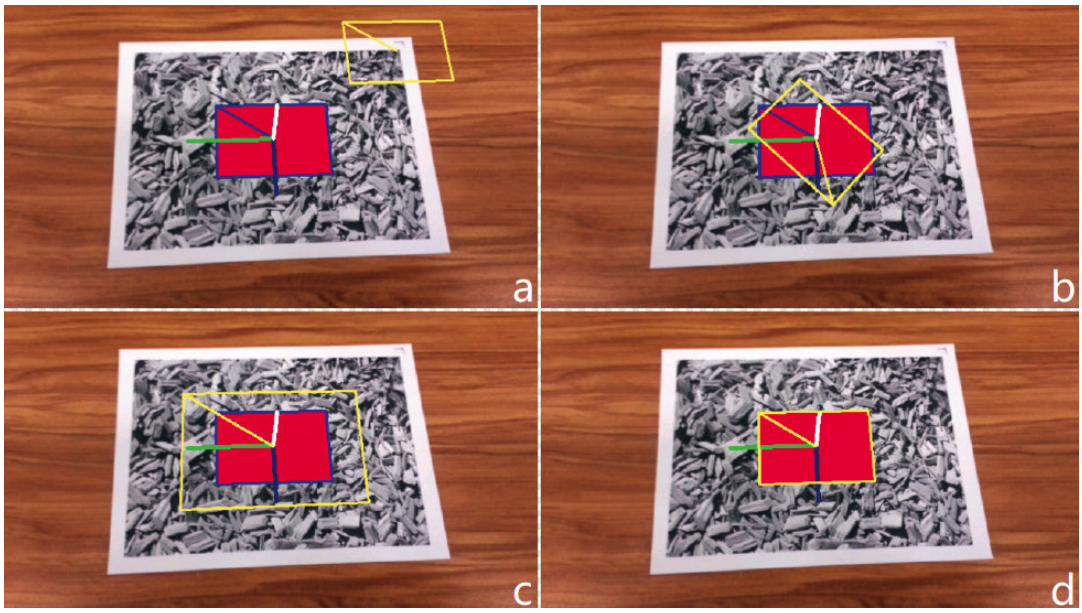

FIGURE 3.10: Screenshots of experimental task samples: target and the manipulated object is the red rectangle with axis attached in the centre, and the target pose and position is the yellow rectangle.

The square would be reset to initial pose and position (Figure 3.10d) after each sub-test was completed. Subjects were told to perform the task as fast and accurately as possible. The order of tasks and sub-tests were randomized for each participant to avoid any order-related influences on the results. Before doing each experimental trial, participants were introduced to the interaction method and they could practice the technique as much as they wanted. After completing the task with each interaction method, participants completed a usability questionnaire and gave further comments at the end of the evaluation.

For the evaluation, we configured the system to automatically measure the performance of participants in terms of task completion time and operation error. The errors were recorded as the value of the distance between the manipulated virtual cube and the target one when the user hit the "*Test Done*" button.

We also evaluated the preference of participants with nine questions (Table 3.2) related to user experience. We used a Likert-scale (1 to 9 with 1 indicating strongly disagree while 9 indicating strongly agree) for each subjective questionnaire item.

3.2.3 Experiment Results

To evaluate the performance records and the results of the user questionnaire, we performed two tests to verify the significance of difference among the three interaction approaches.

TABLE 3.2: Friedman test result

Question	p
Q1. I was performing well	0.211
The given interface was:	
Q2. Easy to learn	0.174
Q3. Easy to use	0.029
Q4. Useful to complete the task	0.025
Q5. Intuitive	0.035
Q6. Natural	0.223
Q7. NOT mentally stressful	0.037
Q8. NOT physically stressful	0.024
Q9. With fun and engagement	0.009

* $N = 10$, $df = 2$.

Using a one-way repeated measures ANOVA ($\alpha = .050$) to analyse the performance measurements (Table 3.3), we found a significant difference among the interaction methods in terms of task completion time ($p < .001$). Participants took more than twice the amount of time to finish all tasks with the gesture based interaction (M = 22.8 sec., SD = 8.8, SE = 0.8) compared to the freeze view touch based interaction (M = 6.4 sec., SD = 2.2, SE =0.2) and the free view touch based interaction (M = 8.7 sec., SD = 3.5, SE = 0.32).

TABLE 3.3: Completion time and operation error of each task

	Free Touch	**Freeze Touch**	**Finger Gesture**
Time			
Translate	8.0(1.9/0.3)	6.3(0.8/0.1)	24.9(9.7/1.5)
Rotate	11.0(4.1/0.6)	7.5(3.2/0.5)	22.0(12.1/1.9)
Scale	7.1(2.9/0.5)	5.3(1.3/0.2)	21.4(11.3/1.8)
Error			
Translate	8.2(4.5/0.7)	5.3(2.9/0.5)	14.8(11.0/1.7)
Rotate	1.7(1.2/0.2)	0.8(0.8/0.1)	6.9(7.2/1.1)
Scale	10.7(6.0/1.0)	12.9(7.7/1.2)	20.1(7.2/1.1)

* Mean (SD/SE). Time values in second; Error values in pixel for translate and scale, degree for rotate.

There was also a significant difference in errors in translation, rotation and scaling tasks between the different interface conditions ($p = .002, .0001$, and $.005$, respectively). The freeze view touch method produced significantly fewer errors than the other two interaction methods especially for rotation and translation.

We used the Friedman Test with an alpha level of 0.05 to verify significant differences in terms of how users felt about the various interface conditions. Table 3.2 presents the detailed p value for each question. There were statistically significant differences between the average questionnaire responses for the three conditions

for the following questions (Friedman test with $p < .04$ for all); Ease of use of the interaction method (Q3), usefulness of using the method for task completion (Q4), intuitiveness (Q5), mental stress (Q7), and physical stress (Q8), as well as fun and engagement (Q9). Figure 4 shows the average Likert scale responses for each condition and each question. Compared with the traditional free view touch, the freeze view touch received higher ratings, while the finger gesture based interaction obtained lower rating in first five aspects of usability questionnaires mentioned above. This discovery is consistent with the result of the post experiment questionnaire for overall rank of method preference for handheld mobile AR interfaces.

In contrast, in terms of fun and engagement, the finger gesture based interaction was considered as the most interesting method, while the freeze view touch was ranked last. However, there was no significance difference found in ratings on the performance confidence, easiness of learning the interaction methods, and naturalness. Figure 3.11 indicates the detailed results of each item.

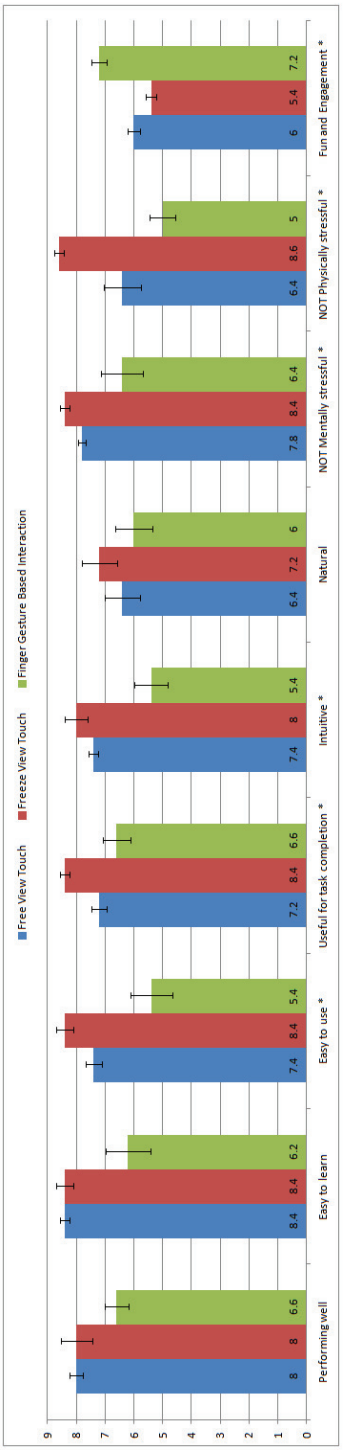

FIGURE 3.11: Usability questions and results: Performing well, Easy to Learn, Easy to Use, Useful, Intuitive, Natural, NOT Mentally Stressful, NOT Physically Stressful, Fun and Engagement (Error bar: +/- SE). The items with significant differences are indicated by using an asterisk.

3.2.4 Discussion

Although the freeze view touch performed fastest and most accurately, users felt that the static AR view considerably reduced the real-time engagement and so was not fun for them compared to the other interface conditions.

The gesture-based interaction appeared to have less usability compared to the other two touch-based interaction methods. This seems to be partly due to the immatureness of our gesture recognition software, as well as the extra time consumed on locking and unlocking the target's editable state, which was included in the task completion time. If a target is located far away from the initial position in the translation tests, users may repeat the unlock-manipulate-lock cycle multiple times because they have to complete several translations to move the object to the final position. In this case, the gesture-based interaction method may take longer than the touch-based interaction technique, which may affect the results. Taking these times out of the consideration, the completion time of our gesture-based interaction method would be reduced by approximately 22.0% on average (to 22.8 sec).

However, it is interesting that the three interaction methods did not show significant differences in terms of performance confidence, easiness of learning the interaction method, and naturalness. This gives an indication that the users felt that the gesture-based interaction is as easy to learn as traditional methods, and it was thought to be as natural as the other methods.

The post experiment questionnaires indicate that interaction based on touching is much easier to use due to users feeling less physical stress. In comparison, users felt that the gesture based interaction was more physically stressful since it requires the participant to move their finger in a larger 3D space instead of a small 2D surface, which could lead to fatigue over time. After a while during the test, most users would place their both elbows on the desk while holding the mobile phone with their hands to interact with it. One user mentioned that "I felt more conformable to use the phone in this pose, and felt easier to keep the interaction more stable, otherwise my arm started aching and got tired really soon". This pose worked pretty well to reduce the fatigue issue in most tasks based on the participants' feedback.

3.2.5 Conclusion

In this study, we investigated two different handheld interaction methods; 2D markerless fingertip-based interaction and freeze view touch. We described how each of these methods was implemented and presented results from a user study comparing these two techniques with each other as well as with the traditional free view touch

method in a mobile AR environment. The user evaluation showed that the freeze view touch method seems beneficial compared to the traditional touch input, but more work is required on the gesture based interaction method before it is as accurate enough, even though users found the gesture input method very enjoyable. Although the sample size is small, our results indicate a trend and are in line with what we would have expected based on the literature. It is though noticeable that new interfaces often perform worse because they are unknown, but users often consider them "*fun*", "*cool*", or "*different*", because they are new and thus exciting. These trends might reverse in a longer, more detailed study.

3.3 3D Markerless Fingertip-based Interaction

Conventional 2D touch-based or fingertip-based interaction methods for mobile AR cannot provide intuitive 3D interaction due to a lack of natural gesture input with real-time depth information. To overcome this, we used an external RGB-Depth camera to extend our 2D fingertip interface into 3D space, and developed a natural 3D markerless fingertip-based interaction technique for mobile AR devices. This allowed the user to perform 6DOF manipulation directly with AR objects using their bare hands in the mid-air. To simplify the recognition, we currently only consider full hand detection without dealing with the problem of occlusion. In the following sections, we introduce our markerless 3D fingertip-based gesture interaction design, implementation and the user study in more detail.

3.3.1 Interaction Design

Object Selection

We apply the idea of distinguishing between a "*click*" and "*hovering*" to implement the object selection procedure. To make a "*click*", we use a pinch-like gesture with the thumb and index finger to select a virtual target, which is very similar to how people grasp real objects using two fingers. When the midpoint of the two detected fingertips is completely inside the geometric space of the virtual object, the virtual object is selected and becomes a candidate for further possible manipulations. A selection is cancelled by using a dwell timing method just like the "*hovering*": compared with the previous state, keeping the midpoint of two fingertips relatively still inside a tiny space region (10 by 10 by 10 mm in our case) around the current position longer than a certain time (2 seconds in our case), will cause the object selection state to be cancelled.

Canonical Manipulation

The RGB-Depth camera is used so that we can retrieve the depth image synchronously instead of only the RGB frame generated from a normal mobile camera. With this we can extract the 3D position coordinates of the detected fingertips and project them into the AR marker's world coordinate system, which is also shared by the virtual object. Thus, we can directly use the absolute position of the two fingertips' midpoint in space as the input value to complete the translation. We also define the rotation input as the absolute 3D pose of the line connecting the user's thumb and index fingertips in space. Finally, we use a two-finger pinch-like gesture and the distance between the two fingertips as the scaling input.

3.3.2 3D Fingertip Detection

In this section we explain the 3D fingertip detection implementation, including hand region segmentation, hand skeleton retrieval and fingertip identification, and describe our prototype hardware and software configuration in detail. With this implementation, the mobile AR system can detect the positions and movements of the user's fingertips, and map these gestures onto corresponding manipulations of the virtual objects in the AR scene. The result is that the user can move, rotate or pinch fingers in 3D space in front of the camera for basic operations.

Hand Region Segmentation

Colour and depth frames retrieved from the RGB-Depth camera are initially calibrated to establish a pixel-to-pixel geometry mapping between them.

To achieve pixel to pixel spatial synchronization between colour and depth frames retrieved from the RGB-Depth camera, we used the UV map method offered by the OpenNI 2 library[7]. We can get the depth data of any pixel on the colour frame, as shown in Figure 3.12 (bottom-right). We optimized the above mapping implementation, and have successfully boosted the frame alignment speed from 12 FPS up to 25 FPS (frame grabbing speed is 28 FPS), which is fast enough for further AR tracking processing in real-time.

For the mobile AR system, we assume that the user's hand interacts with the virtual scene at a distance of between 150 mm and 450 mm away from the mobile camera. With this assumption, we can remove the background from the colour image frames by filtering the colour pixels that have corresponding depth pixels located outside of our assumed depth range. Thus, we can avoid the noise caused by a

[7] http://structure.io/openni/

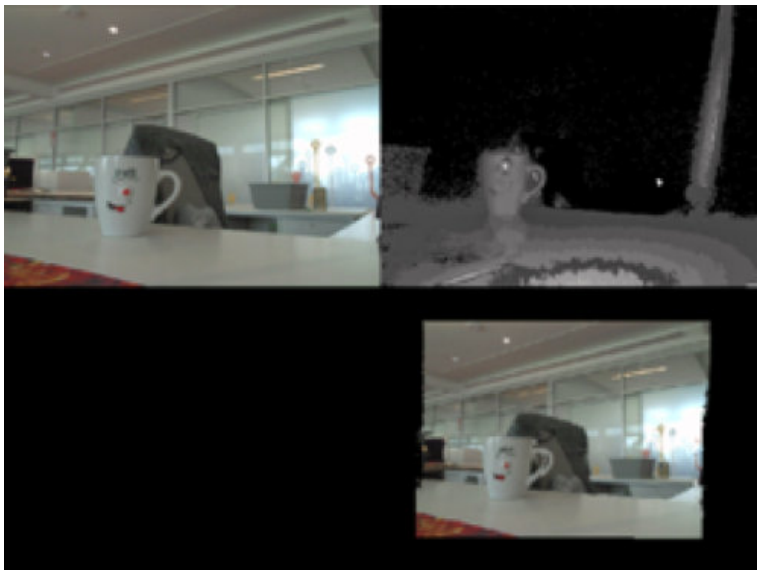

FIGURE 3.12: Bottom-right: mapped color frame that is aligned with the top-right depth frame.

background with a similar colour to the human skin, such as the brown table in Figure 3.13a, and get the filtered depth foreground result in Figure 3.13b.

We also use a skin-color filter to refine the hand region from the mapped colour foreground based on a generalized statistical skin colour model used in Handy AR. We then apply a distance transformation to find the largest single connected component which is considered to be the hand region (Figure 3.13b). To guarantee a smooth hand contour, a Gaussian blurring method is utilized, and a clean hand contour is detected (see Figure 3.13c and Figure 3.13d). With the benchmark hardware described in Section 3.3.3, the whole segmentation process takes around 10 msec for a RGB-Depth frame with 320 by 240 pixel resolution in average.

Retrieving the Hand Skeleton

In this system, we use the hand skeleton to identify the hand movement and locate the fingertip coordinates. Our method uses several steps to retrieve a hand skeleton from the hand contour that we get from the previous output (Figure 3.14a). First we apply a distance transformation to detect ridges in a distance map (Figure 3.14b) of the hand contour. The rows in the distance map are analysed from left to right, comparing the value of the current pixel with its neighbouring pixel to the right. If the neighbour has a greater value then the analysed one will return +, if it is the same value it will return 0, and if the value is smaller then it will return -. The patterns +-, +0- and +[0...]- indicate the ridges of the distance map, which we mark as the skeleton points. This procedure is repeated four times with different analysing directions from

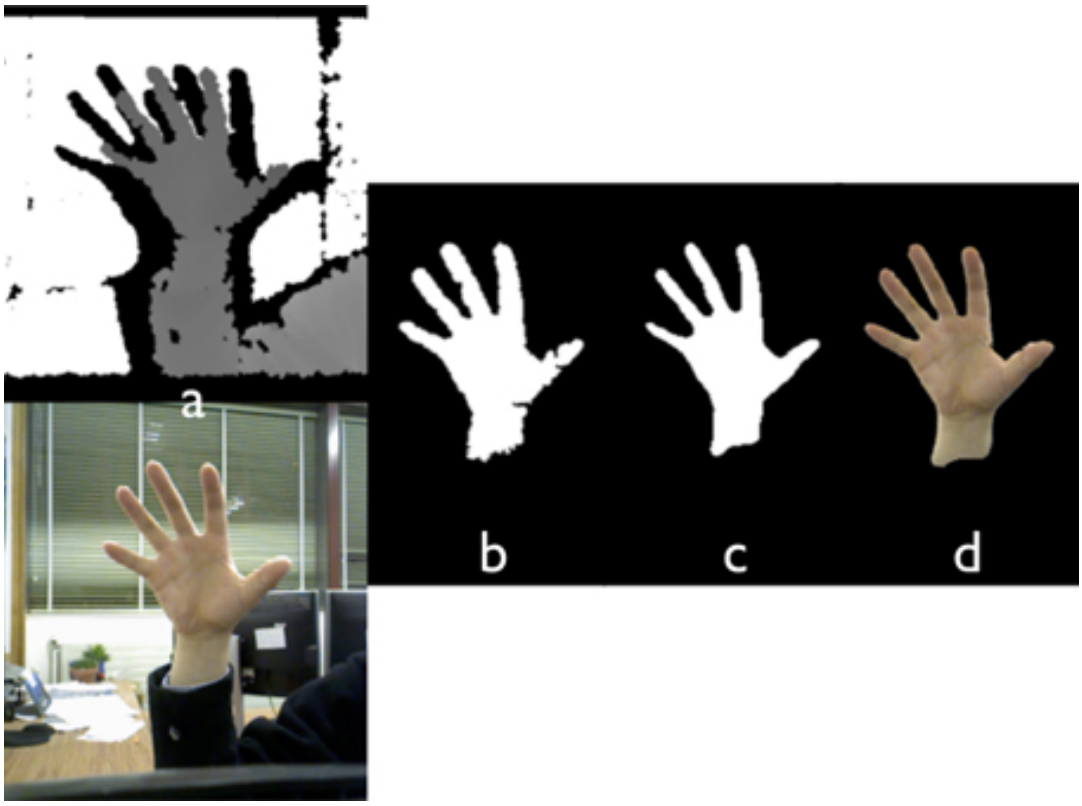

FIGURE 3.13: a. Color and depth frames; b. Foreground depth region; c. Smoothed hand contour; d. Final hand region.

left to right, top to bottom, the top left corner to bottom right corner and from the top right corner to bottom left corner. All the skeleton points detected by these four procedures are merged together to form the skeleton of the hand region (Figure 3.14c).

With the benchmark hardware described in Section 3.3.3, the whole skeletonization process takes less than 5 msec on average for a RGB-Depth frame with 320 by 240 pixel resolution. The above distance transformation based skeletonizing method ensures accurate localization and real-time processing. However, it does not guarantee connectivity of the skeleton, and has obvious noise – being not one pixel width. To deal with these issues, we first use Zhang and Suen's method [118] to erode the skeleton width to one pixel (Figure 3.14d), and then connect the skeleton gaps using Chang's technique [21]. From Figure 3.14d we can observe that the skeleton is separated into several lines. For each endpoint of these skeleton lines, we know that it only links to one existing skeleton point from 8 neighbours around it, and the point with the maximum distance value among the remaining 7 neighbours is selected as the next candidate point and marked as a new skeleton point. This process is iterated from the new candidate point until it reaches one of the existing skeleton points or the boundary of the hand region. Using this process, the isolated skeleton lines are linked into a cohesive whole as presented in Figure 3.14e.

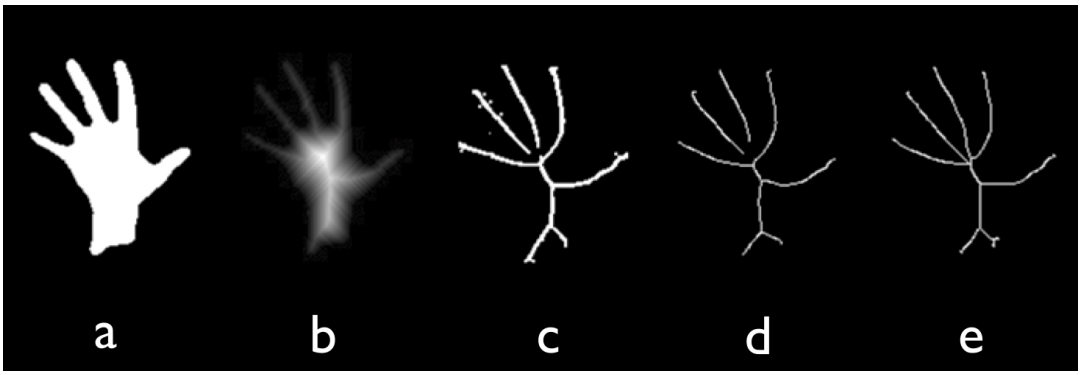

FIGURE 3.14: a. Clean contour; b. Distance transformation; c. Skeletonization from the distance map; d. Skeleton with a single pixel wide; e. Refined skeleton without gaps.

Fingertip Identification

To interact with the AR virtual objects overlaid on the real world, we project the 2D hand skeleton into the 3D space with the help of the depth map (Figure 3.15a), and then we can check the endpoints of the 3D skeleton to identify the fingertips (Figure 3.15b). For normal hand gestures, the fingertips and the wrist are facing the opposite directions from each other from the viewpoint of the palm; therefore, the end points of the skeleton can be divided into two groups, the fingertips and the wrist ends. This classification can be achieved by measuring their distances to the depth camera.

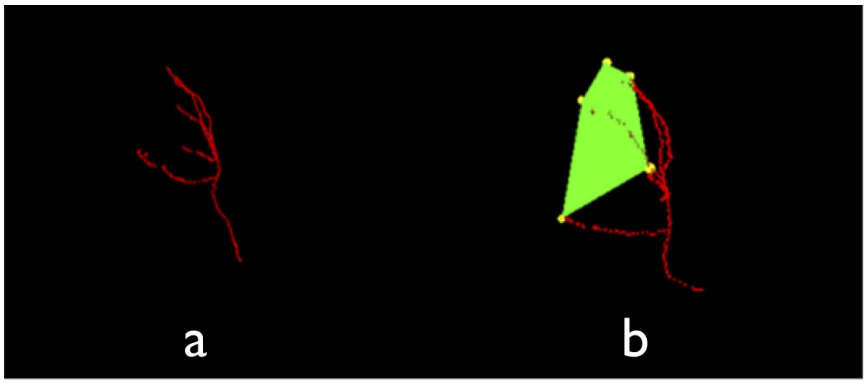

FIGURE 3.15: a. 3D Skeleton; b. Projected fingertips.

3.3.3 Prototype Implementation

Due to hardware limitations during the study, we chose a tablet with a RGB-Depth camera attached to build our prototype. The prototype runs on a Samsung XE500T1C tablet with a Windows 7, featuring an Intel 1.8 GHz Core i5 CPU and a 4 GB RAM. All calculations were completed using CPUs with multithreading technology but without

any GPU working load. A DS325 from Softkinetic[8] has been used as the external short range RGB-Depth camera. The sensor has an operational range of between 150mm and 1000 mm with a depth error of less than 14 mm at 1 m, which makes it is easy for users to locate their hands from the tablet within at such a distance. This provides spatial synchronized colour and depth frames with 320 by 240 pixel resolution at 30 FPS to the tablet via a standard USB connection. We attached the sensor directly on the back of the tablet for our prototyping (Figure 3.16).

FIGURE 3.16: System prototype: the tablet with a RGB-Depth camera attached on the back.

We used the Softkinetic driver to obtain image data from the depth sensor, combined with the OpenCV[9] library for colour-depth pixel alignment, and rendered a virtual scene in our system using OpenGL[10]. AR tracking was implemented using a natural feature-tracking library called OPIRA [25]. We choose this library due to its feasibility, robustness and fast computation time on the Windows OS. Furthermore, OPIRA also has a convenient interface for integrating with OpenGL. Our system ran up to 20 FPS and could be optimized to speed up the performance further in the future.

Figure 3.17 demonstrates a screen shot of our running system, where the red lines indicate the retrieved hand skeleton, the yellow dots indicate the detected fingertips. The picture also shows the AR scene in which a green virtual cube is overlaid on the top of the image marker. This cube can be grabbed by the hand and moved in 3D AR space without any extra touch UI or IMU sensor support.

[8] http://www.softkinetic.com/
[9] http://opencv.org/
[10] http://www.opengl.org/

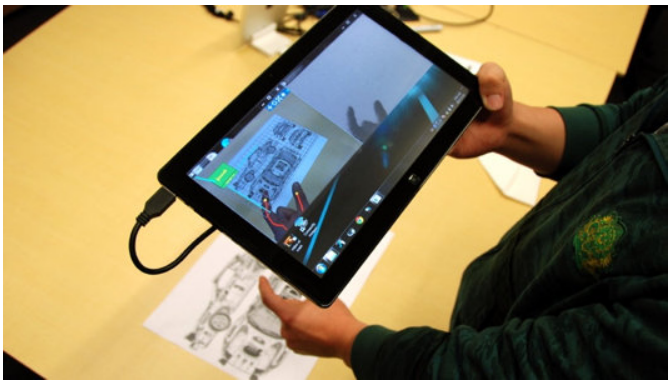

FIGURE 3.17: Using the fingertip detected from the 3D hand skeleton to move the augmented cube.

3.4 User Study

To investigate the performance and usability of our prototype 3D gesture interface, we conducted a user study comparing gesture input with a traditional touch approach across three fundamental scenarios with varying tasks.

3.4.1 Experiment Setup and Procedure

We set up the user study using a within-group factorial design where the independent variables were the manipulation technique and task scenario. The manipulation techniques are our prototype 3D fingertip-based interaction method and traditional 2D screen-touch input, while the test scenarios contain three different experimental tasks with varying subtasks. The dependent variable is task completion time and we also measured user preferences for both techniques in terms of usability.

To begin the study, each participant was asked to complete a pre-test questionnaire about age, gender and prior experience in touch-based mobile devices, 3D gaming interfaces and AR applications. A brief introduction to our mobile AR system was then given to the participant, followed by a detailed instruction of the 2D touch and 3D gesture manipulations used in our testing environment. The participant learned the general operation attention, basic interface usage, and overall task content. Afterwards, each participant had ten minutes to practice both interaction techniques. Once they started the study, they were not interrupted or given any help. Upon the completion of the practical task with each interaction method, they were asked to fill out a per-condition questionnaire (Table 3.4) and gave further comments on a post-experiment questionnaire (Table 3.5) at the end of the evaluation. The whole user study took approximately 45 minutes on average.

TABLE 3.4: Per-condition questionnaire

Question
Q1. Easy to learn
Q2. Easy to use
Q3. Natural
Q4. Useful to complete the task
Q5. NOT mentally stressful
Q6. NOT physically stressful
Q7. With fun and engagement

For the evaluation, we collected the user preference with seven questions related to the usability in Table 3.4, on a Likert-scale (1 to 9 with 1 indicating strongly disagree while 9 indicating strongly agree) for each subjective questionnaire item. Furthermore, we configured our testing system to automatically measure the task completion time of the participants.

TABLE 3.5: Post-experiment questionnaire

Q1	Which interface do you prefer to use if you will have to do a similar task again?
Q2	When determining how much you like using a manipulation technique for mobile AR? How important in influence on your decision was ease, speed and accuracy?
Q3	Please briefly explain the reason you chose the interface above.
Q4	Please briefly explain the reason you chose the interface above.
Q5	Any other comments on the interface or the experiment?

A total of 32 participants (16 male and 16 female) were recruited from outside of the university for the experiment. Their ages ranged from 17 to 52 years old (M = 35.03, SD = 10.28). All participants were right-handed. During the experimental tests, 29 participants held the device in their left hand and used the right hand for interaction, while for the other three it was the other way around. No significant differences could be observed regarding handedness. All of them used the index finger for touch input and extra thumb finger for gesture manipulations. Although 27 of them used touch screen devices frequently, only six of them had some experience of using 3D interfaces, mainly from the game consoles like Microsoft Kinect[11] or Nintendo Wii[12]. None of them had previous experience with using AR interfaces.

[11] http://www.xbox.com/kinect/
[12] http://www.nintendo.com/wii/

3.4.2 Tested Scenarios

Based on the work of Bowman [18], we used the basic canonical manipulation tasks of translation, rotation, and scaling in the task design, and built three specific scenarios with several subtasks to cover typical 3D manipulation situations in handheld mobile AR applications. To manually identify the desired object for subsequent manipulation, the canonical operation of selection was also used.

Each participant was asked to perform three experimental tasks using the two interaction interfaces (traditional 2D touch and novel 3D gesture interaction) respectively, and each task with each interface was tested twice for repetition. The interfaces were presented in a random order to the participants to exclude potential learning effects. One additional selection and three types of essential manipulations (translation, rotation, and scaling) were included in the tasks for each interface test. The order of tasks and related sub-tests were randomized for each participant to avoid any order-related influences on the results.

The experimental task was to select and manipulate a virtual cuboid in a mobile AR application and match it to the indicated target position, pose or size. For all tasks, our system set the target colour to blue, green for the object the participant can control, and red for the selected object which is currently being manipulated. All test tasks were presented in the same virtual AR background environment. A black and white textured plane printed on a 297 by 420 mm piece of paper was the target marker for the AR tracking. Meanwhile, both the indicated target and manipulated object were clearly displayed on the same screen, and the participant could inspect the scenario from a different perspective by moving the tablet freely to understand the task before officially conducting the actual test (Figure 3.17).

Subjects were told to perform the task as fast and accurately as possible. The timing of the task was automatically started after the virtual object was successfully selected, and was stopped automatically when the task was completed, which means that the cuboid was changed to the required status in the space by the participant's operations, so no error measure need to be conducted.

3.4.3 Experiment Results

To analyse the performance time and the user questionnaire, we performed a Wilcoxon Signed-Rank Test using the Z statistic with a significance level of 0.01. The Dependent T-Test for analysing time performance was not applicable because the data sets that we collected were not normally distributed.

TABLE 3.6: Wilcoxon Signed-Rank test

	Translation		Rotation		Scaling		Selection	
	Z	p	Z	p	Z	p	Z	p
Q1	-3.532	0.000412	-3.500	0.000465	-2.887	0.003892	-4.973	0.000001
Q2	-1.170	0.241887	-4.476	0.000008	-3.900	0.000096	-5.064	0.000000
Q3	-1.057	0.290578	-1.615	0.106209	-1.155	0.248213	-4.291	0.000018
Q4	-0.943	0.345779	-4.233	0.000023	-2.646	0.008151	-4.640	0.000003
Q5	-1.000	0.317311	-1.732	0.083265	0.0001	1.000000	-2.828	0.004678
Q6	-3.750	0.000177	-4.409	0.000010	-4.455	0.000008	-5.055	0.000000
Q7	-4.666	0.000003	-4.821	0.000001	-4.911	0.000001	-4.816	0.000001

Task completion time was the main criterion for performance evaluation in our test, which represents the time for the subject to complete an entire canonical task. Analysing the data from the performance measurements, we found a significant difference between two interaction methods in terms of overall completion time ($z[n : 32] = -4.862$, $p < .0001$). On average the tasks were performed significantly faster with 2D screen-touch interaction. When inspecting the mean completion time for each manipulation task, significant differences could also be found for translation ($z[n : 32] = -4.020$, $p < .0001$), rotation ($z[n : 32] = -4.937$, $p < .0001$) and scaling ($z[n : 32] = -4.619$, $p < .0001$). Subjects took more time (around 50 sec.) to finish all tasks with the 3D gesture-based interaction compared to the 2D touch-based interaction.

Figure 3.18 shows the average response to each of the subjective questions for the two conditions. Analysing these results using a Wilcoxon Signed-Rank Test we found the statistical values shown in Table 3.6. The results reveal that users thought that the 3D gesture was more enjoyable to use, and no significant difference was found between using 3D gesture or 2D touch input in terms of naturalness and mental stress.

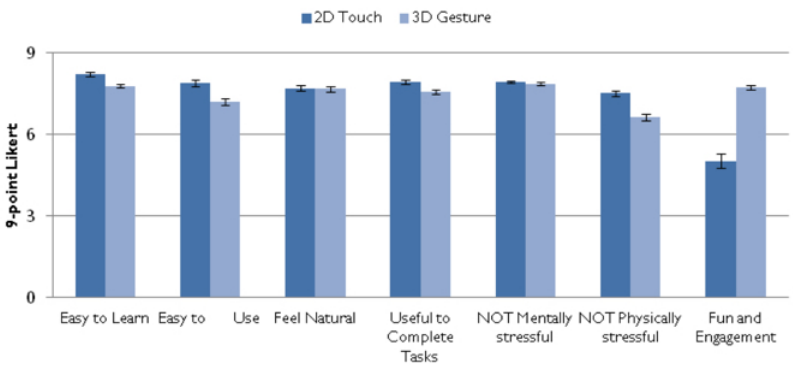

FIGURE 3.18: Users' average rating for two interfaces.

Figure 3.19 shows the subjective questionnaire results following the translation manipulation task. We found significant difference in terms of ease-of-learn (Q1) ($z[n : 32] = -3.532$, $p < .0005$), not-physically-stressful (Q6) ($z[n : 32] = -3.750$, $p < .0002$), and fun-and-engagement (Q7) ($z[n : 32] = -4.666$, $p < .0001$). Users felt that when performing translation tasks, the gesture-based interface was not as easy to learn as the touch one, was physically more stressful to some extent. In contrast, they felt gesture based interaction was more fun and engaging than touch input. The gesture translation was also ranked higher in terms of its naturalness, but not significantly so. There were also no significant differences found for all of the other four questions in the translation task ($p > .2$ for all responses).

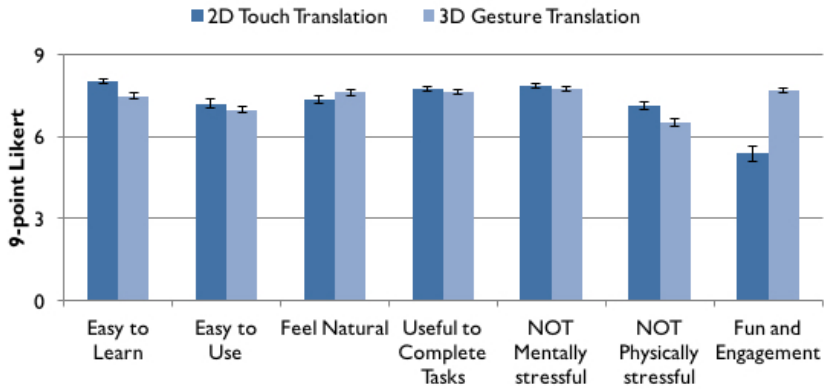

FIGURE 3.19: Usability results for translation.

Figure 3.20 shows the subjective questionnaire results following the rotation manipulation task. We found significant difference in terms of ease-of-learn (Q1) ($z[n : 32] = -3.500$, $p < .0005$), ease-of-use (Q2) ($z[n : 32] = -4.476$, $p < .0001$), useful-for-tasks (Q4) ($z[n : 32] = -4.233$, $p < .0001$), not-physically-stressful (Q6) ($z[n : 32] = -4.409$, $p < .0001$), and fun-and-engagement (Q7) ($z[n : 32] = -4.821$, $p < .0001$). Similar to the translation results, gesture based rotation method was not thought to be more easy to learn or less physically stressful than touch interaction. In addition, it was considered not so easy to use, and not so useful for the task completion compared to touch input. However users did feel that the performing rotations with gesture input was more fun and engaging. There was no significant differences found for the questions about naturalness (Q3) ($z[n : 32] = -1.615$, $p > .1062$) and metal stress (Q5) ($z[n : 32] = -1.732$, $p > .0832$).

Figure 3.21 shows the subjective questionnaire results following the scaling manipulation task. From table 5.4 we can see there was a significant difference on most questions except the naturalness (Q3) ($z[n : 32] = -1.155$, $p > .2482$) and not-mentally-stressful (Q5) ($z[n : 32] = -0.000$, $p < .9999$). As illustrated in Figure 5.4,

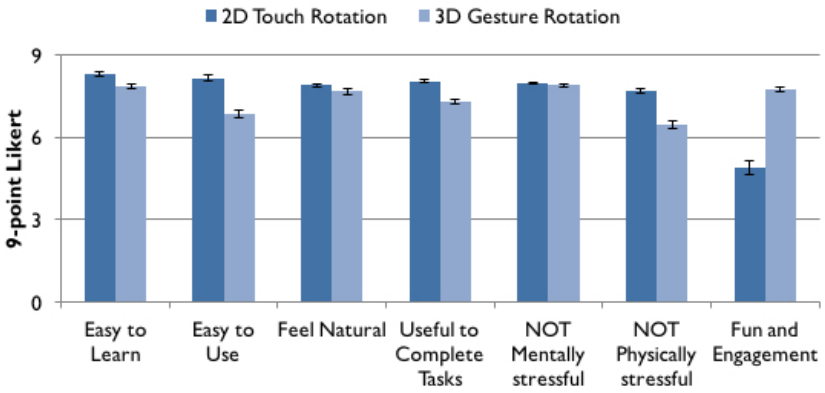

FIGURE 3.20: Usability results for rotation.

users felt that the gesture-based method was as natural and non-mentally-stressful as the touch interface. However, participants felt the gesture interface was more physically stressful to use and it was not as easy to complete the scaling task while using gesture to interact with virtual objects. Similar to the earlier results, gesture interaction was thought to be more fun and engaging.

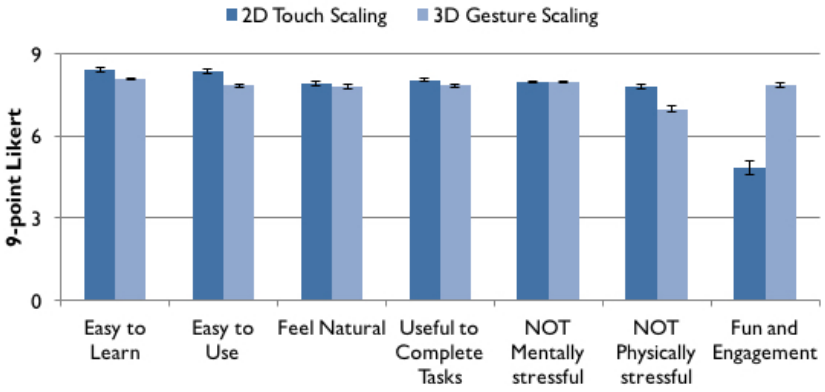

FIGURE 3.21: Usability results for scaling.

In terms of object selection there was a significant difference between conditions across all the questionnaire items ($p < .005$ for all), as shown in Figure 3.22. Looking in detail at users' rating in each question, the 2D touch-based selection was significantly preferred by the subjects except in terms of fun-and-engagement part. This may be because users are not used to selecting virtual items from space by using gestures without being able to feel them.

Reviewing all the results above, we found that subjects gave different ratings to the same questionnaire items for different interaction tasks. However, there was no significant difference between conditions found in terms of naturalness and mental stress for all the manipulation tasks.

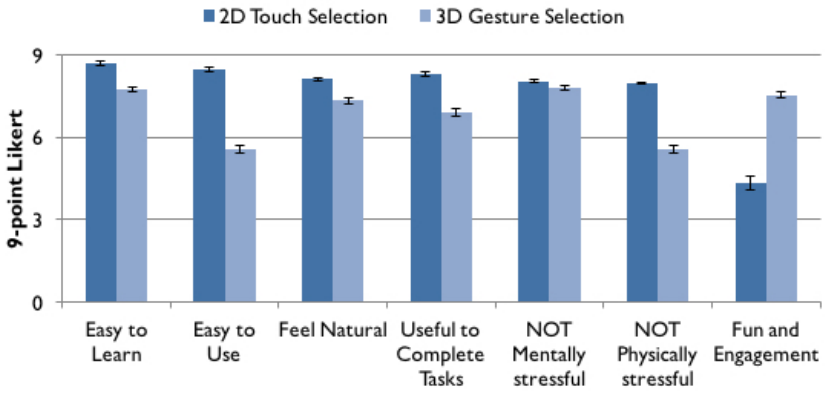

FIGURE 3.22: Usability results for selection.

The post experiment questionnaires indicate that the touching is the subject's first choice. This discovery is consistent with the result of the ranked method preference for the mobile AR interaction.

3.4.4 Discussion

Although our gesture based interaction approach can provide a more interesting and engaged experience for users to manipulate the virtual objects, most participants still prefer to use screen-touch interface to finish tasks. The main reason is that most participants in our research were experienced with using touch input devices such as smart phones; therefore, they get used to touch input and feel comfortable when they use the 2D touch method. In contrast, the 3D gesture technique requires all participants to learn a new technique while the method itself is not so easy to use for the people who have little experience with 3D or AR interfaces.

There are several reasons that might explain why our gesture system is not as attractive as we expected. First, when users use their fingers to select virtual objects in the 3D space, they have no physical touch feedback, which can increase the selection difficulty. Although we obtain the depth information for the fingertips via the depth camera, virtual objects are simply overlaid on top of video frames and always cover the user's fingers no matter how deep they are. This can generate visual confusion and increase operational complexity. To improve the depth perception, we could add vibro-tactile feedback just as Jin and Park's work [50], or use acquired depth data to segment the foreground and background, handling with occlusion correctly between the hand gesture and the virtual objects based on their spatial relationship. In addition, most participants prefer using a pinch-like gesture with two fingers to rotate or scale the virtual target, but they insist on using one finger selection as it is much more straightforward and effective for selecting manipulations to them.

Secondly, after starting the manipulation, users normally hold the tablet in a fixed position while the other hand is stretched out to reach the virtual object; but the interaction hand often moves out of the camera view, which can cause the fingertip position to be lost. This may cause participants to lose their patience in some degree, which cause negative feedback on the coming tasks. Thirdly, when we use two fingers to rotate the target, one finger may cover the other in some certain angles. Meanwhile, participants can only rotate their fingers over a small angle range; when the target has a too large or too small angle, the manipulation task cannot be completed easily and people have to move the tablet to adjust to the best observation spot. To deal with these issues and improve the usability of our 3D gesture interface, a steadier fingertip detection algorithm could be developed in the future.

3.4.5 Conclusion

In this study, we presented 3D gesture-based interaction approach for mobile AR on a tablet with an external RGB-Depth camera attached to it. It is based on identifying fingerprints movements and mapping them into operations on virtual objects. We conducted a user study to evaluate our approach by comparing it with touch-screen based interaction for virtual object manipulation. This evaluation measures performance (time) and engagement (subjective user feedback). The results of this study show that the touch-based interface outperforms the gesture-based interface in terms of performance. However, our method indicates a high entertainment value, suggesting great possibilities for leisure applications but limited usage for time-constrained tasks.

Chapter 4

Free-hand Skeleton-based 3D Interaction

Compared with touch-screen input, the natural gesture-based interaction methods that we studied in Chapter 3 can offer a more enjoyable user-centric experience for mobile AR applications. However, most gesture interaction techniques for mobile AR only use two degrees of freedom, without depth input (see the example in Section 3.1), while AR virtual objects are overlaid on a view of a real three-dimensional space.

In this chapter, we present a client-server framework for a markerless gesture-based 3D interaction method for mobile AR on the smart phone in a small workspace. Compared with our previous interaction technology described in Section 3.3, the new system delivers a 3D skeleton-based interaction implementation with depth info included on a smartphone with embedded OS (Android in our case) instead of the Windows tablet. We describe the implementation process of each step in detail and present performance results of markerless fingertip-based prototypes in Section 4.1 and skeleton-based prototypes in Section 4.2 respectively. Finally we conduct a formal user study in Section 4.4 for the skeleton-based 3D input method to evaluate the usability of its canonical manipulations by comparing it with touch-based techniques. Later, in Chapter 5 we explore in three more applications developed based on this framework.

4.1 Finger-based 3D Interaction within a Client-Server Framework

Inspired by the research of Piumsomboon et al. [89], we investigate a novel mobile AR interaction method based on a client-server framework. This framework allows users to perform 3D manipulations on virtual objects with their fingers in mid-air.

4.1.1 Client-Server Framework Design

We setup the whole system in a small workspace where the desktop computer server and the mobile phone client share a same interaction volume, as shown in Figure 4.1.

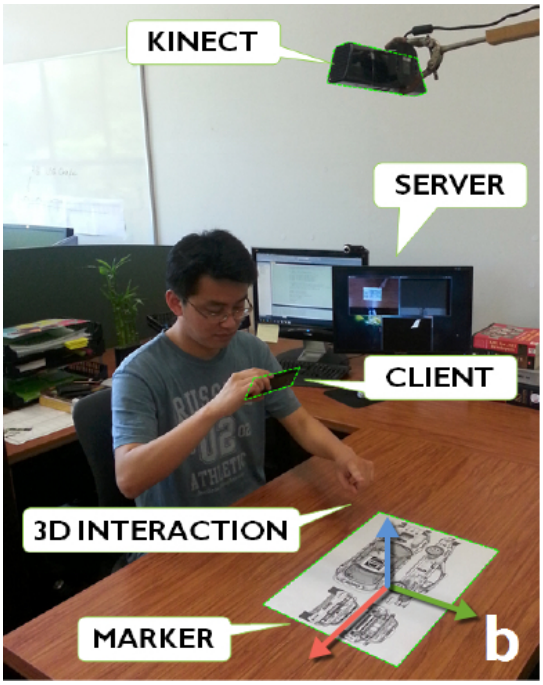

FIGURE 4.1: Client-server framework with gesture input for mobile AR.

A PC connected with the depth camera is used as a server, and a smartphone with a single built-in colour camera is used as a client. On the server PC we use a depth camera and gesture software to segment the hand and identify fingertip 3D positions in real time. We then wirelessly transmit the fingertip data to the smartphone, mapping mid-air gesture commands onto the selected virtual object shown on the mobile screen.

A Microsoft Kinect (RGB-Depth camera) is positioned 80 cm above the desired interaction space facing downwards, and a printed reference image marker is placed on the table right below the camera. The OPERA on the server (Section 3.3.3) and the BRISK-based method on the client (Section 3.1.1) track the position of the Kinect camera (RGB lens only) and the built-in client camera relative to the same reference marker. Thus, the two tracking systems use the same World Coordinate System (WCS), the origin of which is located in the middle of the AR image marker (Figure 4.1). Meanwhile, the gesture software (described in Section 3.3.2) on the server uses the Kinect camera to generate fingertip positions in the Camera Coordinate System (CCS) and project these values into the WCS synchronously. After that, these 3D values detected from the server side can be directly transmitted to the client via

wireless data communication, and can used by the mobile client for the final 3D interaction input without any further coordinate system transformation.

Thus, the system eventually support real-time 3D finger-based interaction for video-based AR applications on mobile devices. The user can hold the smartphone with one hand while interacting with AR scene with the other hand's fingers.

4.1.2 Prototype Implementation

The RGB and depth outputs from the Kinect are calibrated by Kinect itself to enable a mapping between them, using the same method described in Section 3.3.3, and then these two sources are delivered to the server for further image processing. The results are wirelessly transmitted to the mobile client for mobile AR 3D interaction. The detailed process within the framework is illustrated in Figure 4.2 and the description of each component is as follows.

1. Fingertip detection from RGB-Depth frame

We used the same implementation described in Section 3.3.2 to get the 3D fingertip position from the RGB-Depth image frames on the server. However, it is noticeable that the fingertip's depth that we defined is not the point value in the corresponding coordinate of the depth frame mapping from the RGB frame, but the average depth value of a small nearby region, which circularly fits the fingertip curvature while excluding the non-skin part. This can critically avoid an invalid depth value of a single fingertip point, which could often be located in the shadowing of the infrared area in the depth frame without any actual depth value.

2. Marker tracking

We used OPIRA, a NFT method mentioned in Section 3.3.3 to track the image on the server PC, and used the same BRISK-based NFT method that we developed in Section 3.1.1 to track the same image marker on the mobile phone.

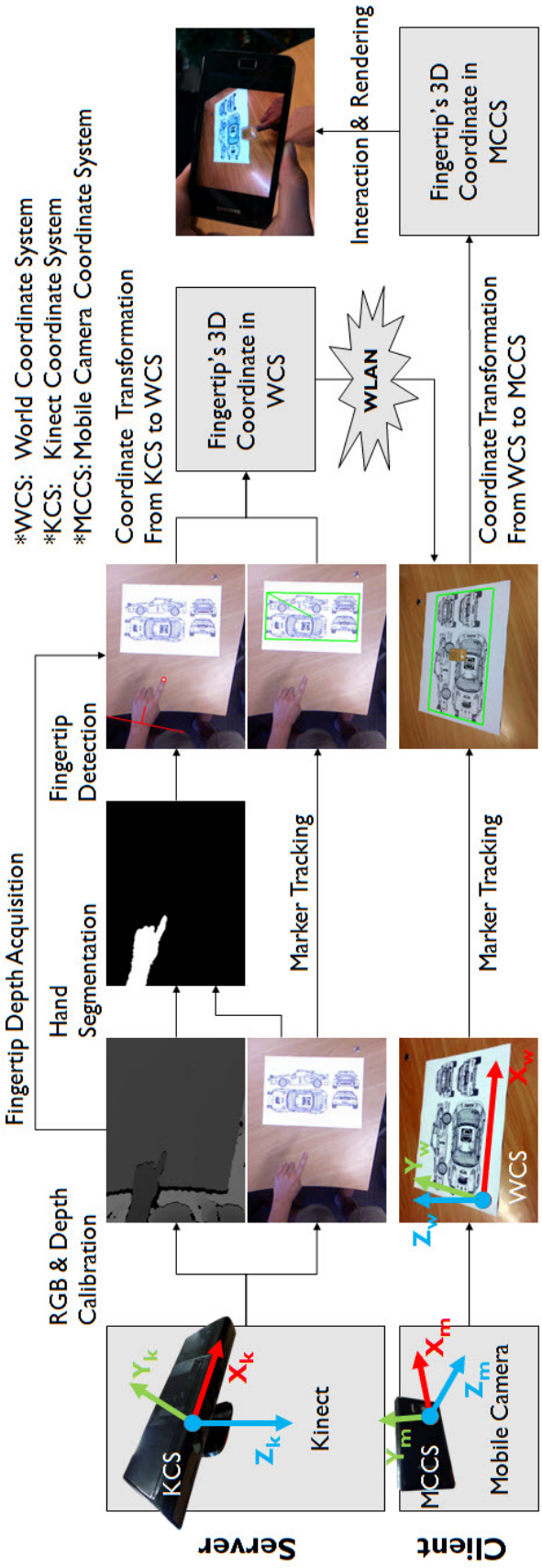

FIGURE 4.2: System workflow in details.

3. Coordinate transformation

The system projects the fingertip 2D position and depth in the image marker's WCS by using the homography matrix obtained from the server tracking session, and then sends this data to the mobile client. The mobile AR application reprojected the fingertip 3D position back to the mobile CCS for any visualization on the screen based on the homograph matrix calculated from the mobile tracking session.

4. Data communication

We used a socket library Asio[1] both on the desktop server and the mobile client for real-time User Datagram Protocol (UDP) based communication. The data communication is fast and stable enough with only a slight delay (less than 20 msec) over our private testing network.

5. Gesture interaction

Three atomic operations (translation, rotation, scaling) can be used to manipulate virtual objects with finger gestures. We mapped the virtual object translation value onto the position change of a single fingertip movement (Figure 4.3). The rotation and scale values are controlled by the orientation and distance change between two fingertips in mid-air respectively.

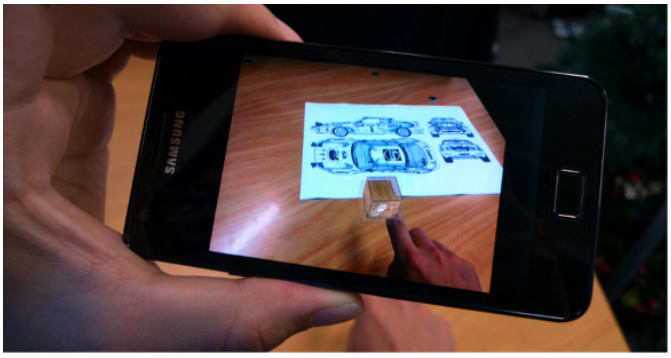

FIGURE 4.3: The user holds the mobile phone with one hand, while stretching out the other and using one fingertip to interact with a virtual box to lift up it from a low position to a higher one in 3D space. In this translation case, the depth value of the fingertip is used.

4.1.3 Performance

We tested the server on a desktop PC (Intel 2.4 GHz Dual-Core i5 CPU, 4 GB RAM, Windows 7) and the client on the same Samsung Galaxy S2. Our prototype ran robustly on the target phone at an interactive frame rate of around 15 FPS with 640 by 480 pixel camera resolution, while the network delay between the server and the client averaged 20 msec. The Kinect provided an error of less than 25 mm when

[1] http://think-async.com/

placed around 800 mm from the marker plane. The manipulation accuracy of our system was within 1cm in the physical WCS. This indicates that the prototype is not suitable for the mobile AR applications that need high-precision input, but it meets basic requirement of general interaction tasks.

Although depending on an external depth camera on the server, this framework prototypes the type of interactions that will be possible in the near future when depth sensing becomes widely available on mobile AR devices. In the future, we will focus on improving the natural gesture recognition to enhance the usability.

4.2 Skeleton-based 3D Interaction within a Client-Server Framework

There are some research projects that have implemented full-fledged 3D hand tracking with the help of depth cameras or customized sensors [55] [116] [100]. However, they are not feasible to be directly integrated into a mobile device because of the complexity and heavy computational cost of the gesture recognition algorithms used. We updated the previous finger-based system to a novel system that provides full-hand skeleton-based interaction. With this system, more complicated gestures can be defined and recognized as inputs without problems with gesture occlusion. For example, Figure 4.4 shows how the user can do a pinch gesture with the thumb and index finger to select a virtual teapot in an AR scene. This section provides more detail about how this interaction system works.

FIGURE 4.4: The user is interacting with a virtual teapot using free-hand input on the mobile device. A virtual hand skeleton is textured on the real hand image, and an augmented teapot is appearing overlaid on the tracked image.

4.2.1 System Improvement

We setup a similar system as the one described in Section 4.1.1 as shown in Figure 4.5 with some important updates from the following three aspects.

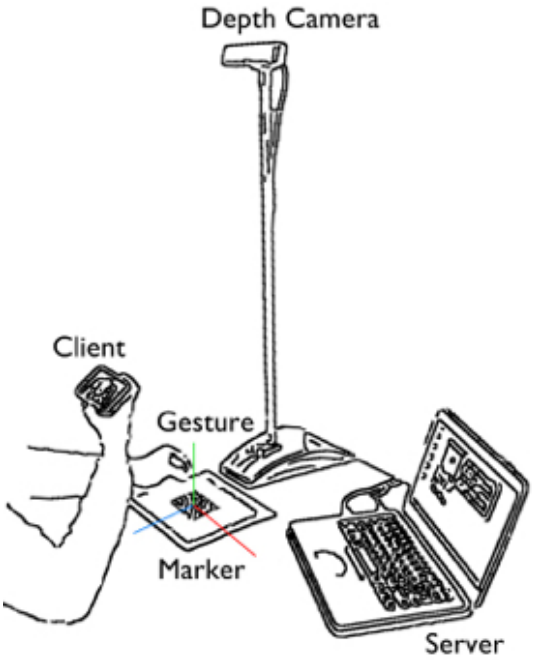

FIGURE 4.5: Updated system setup: The depth camera and the mobile camera share the same coordinate system of the AR marker. The hand gesture can be detected by the depth camera from the PC server side and the skeleton result can be wirelessly transmitted and mapped to AR scene on the phone client.

Hardware

To make our system less bulky and easier to be deployed, we replaced the Kinect with a PrimeSense sensor[2]. The PrimeSense has a much smaller physical size than the Kinect with similar depth accuracy, colour and depth resolution. We also configured the server with a laptop instead of a desktop PC.

Software

We replaced our own 3D fingertip detection solution with the 3Gear software[3] on the server side. The 3Gear software uses the PrimeSense camera to generate on-the-fly 3D models of both hands in the space while using an image marker for the initial calibration as a coordinate reference. We replaced our own NFT method with the Vuforia AR tracking[4] on the client side. The Vuforia software enables stable and robust AR tracking while offering an overall faster and more accurate tracking performance.

[2] http://www.primesense.com/
[3] http://www.threegear.com/
[4] http://www.vuforia.com/

We used Alljoyn[5], a wireless library to enable a reliable data communication among multiply platforms instead of a simple UDP method. The joint and fingertip positions of hands found by the 3Gear software from the server can be directly transmitted to the smartphone in real time via a fast wireless network with Alljoyn.

Coordinate transformation

In this new setup, the server only runs the 3Gear without conducting any AR tracking. We set the Gear hand estimation and tracking software's reference coordinate system to be located in the centre of the image marker. In this case, the value of the joint and fingertip positions of hands detected from the server can be used directly without any transformation computing at all, which will remove one computational step from the whole system, and boost the performance speed in general.

The prototype system allows the user to manipulate virtual objects using pinch-based hand gestures. This consists of pinching at a point of interest then moving the hand for further manipulation (similar to pressing a button on a mouse then dragging), and includes translation, rotation, and scaling. In the translation mode, the selected object follows the pinch position as the user moves the hand, while in the rotation mode it rotates according to the user hand orientation. Scaling is applied uniformly depending on the distance of the user's hand displacement after pinching. In contrast, the traditional touch-based interaction utilizes screen touching instead of mid-air pinching to control the virtual target.

4.2.2 Performance

We ran the server on a Dell laptop (Intel 2.4 GHz Dual-Core i5 CPU, 4GB RAM, Windows 7 OS) and the client on a Samsung Galaxy S3 smartphone (ARM 1.4 GHz Quad-Core Chipset, 1GB RAM, Android 4.1 OS). The hand tracking ran at 25 FPS on the laptop with a fingertip accuracy of around 5 mm. In addition to tracking hand position and pose, the software can also recognize simple hand gestures such as finger pinching. Meanwhile, the Vuforia library tracked at an interactive frame rate of around 30 FPS with 1280 by 720 pixel camera resolution on the target phone, and the communication delay between the server and the client averaged less than 10 msec in our private local area network (up to 300 Mbps). The average speed and accuracy performance was significantly improved compared with the result presented in Section 4.1.3.

This framework allows us to rapidly prototype a mobile AR system that supports 3D hand gesture input. However, one limitation is that the user needs to keep their

[5] http://www.alljoyn.org/

hands within the 3Gear detection volume (110 cm by 65 cm by 40 cm above the image marker in our setup), preventing them from moving outside the interactive space and losing the hand tracking.

4.3 User Study

We conducted a user study using a within-group design to investigate the benefits and drawbacks of our proposed gesture-based interface for canonical manipulations compared to the traditional touch-based interface. The main independent variable was the type of interface (touch and gesture) across three fundamental scenarios with varying tasks. The dependent variables included task performance and participants' ratings collected with preference questionnaires in terms of usability.

We implemented touch interaction to perform similar operations supported by gesture interaction. With touch interaction, users can switch between modes by three finger tapping, while simple tapping will select an object. Dragging selected objects will result in manipulating the object depending on the transformation mode. While one finger dragging only supports 2D transformation on the tracking plane, in our implementation dragging with two fingers allows transformation along the axis perpendicular to the plane.

4.3.1 Experiment Environment and Task

We designed three experimental tasks in a mobile AR application each focusing on different canonical object manipulations: translation, rotation, and scaling.

The first translation task requires the user to move a colorful cube in the lower left corner to overlap a half-transparent red cube located in the upper right corner in all three dimensions (see Figure 4.6a). The distance between the cubes was 115 mm (130 mm by 80 mm on the marker plane) horizontally and 45 mm vertically. In the second task, a cube was placed with a rotation value of 310, -150, 10 degrees around the local x, y, and z axes respectively. The user needs to rotate the cube back to a pose, aligned parallel to the axes on the side while having the blue side facing toward the user and pink side upward, as shown in Figure 4.6b. The third scenario needs the user to uniformly scale an inner colorful cube to approach the size of a half-transparent outer one (see Figure 4.6c). The original edge size of the operational cube was 30 mm and the target dimension was 70 mm. The touch-based scaling method was configured to the uniform scaling mode to keep the comparison with the gesture input as fair as possible.

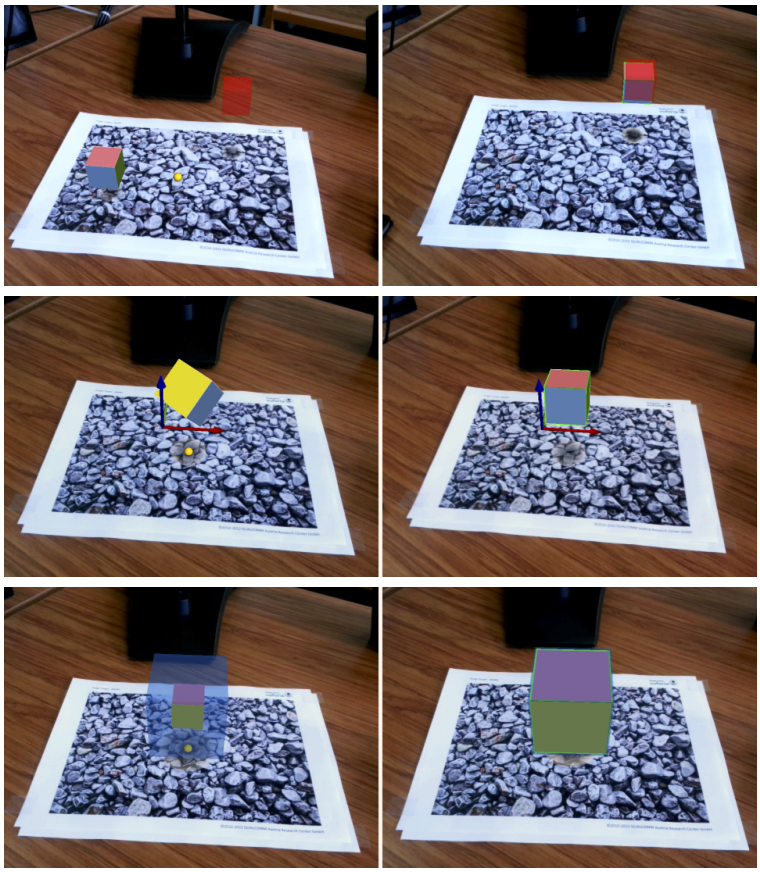

FIGURE 4.6: Three sample tasks: a) Translation, b) Rotation, c) Scaling. The left pictures show the initial status of the virtual objects, and the right pictures are the targeted results that need to be completed by the participants.

We asked users to perform each task by manipulating a virtual object to match the position, orientation, and size of a target object. Participants sat in front of the desk with the setup shown in Figure 4.5, and they were asked to hold the smartphone with the non-dominant hand, so that they could use the dominant hand for touch input or for gesture interaction. They were asked to perform the task as quickly and accurately as possible.

4.3.2 Experiment Procedure and Tested Scenarios

The experiment started with an introduction and participants filling out a pre-experiment questionnaire. Then the participants were given instructions on how to use the prototype system to perform the experimental task. Once they got familiar with the system in the training session (5 minutes for each interface), they proceeded to complete the experimental trials.

The experiment was a within subject design. There were six trials in total with two conditions (using touch and gesture-based interfaces) and three tasks (translation,

rotation, and scaling, as shown in Figure 4.7), while each task was tested twice by the participants. The order of the interfaces was counter balanced by alternating them between participants, but the order of tasks was fixed as the purpose of the experiment was not to compare them.

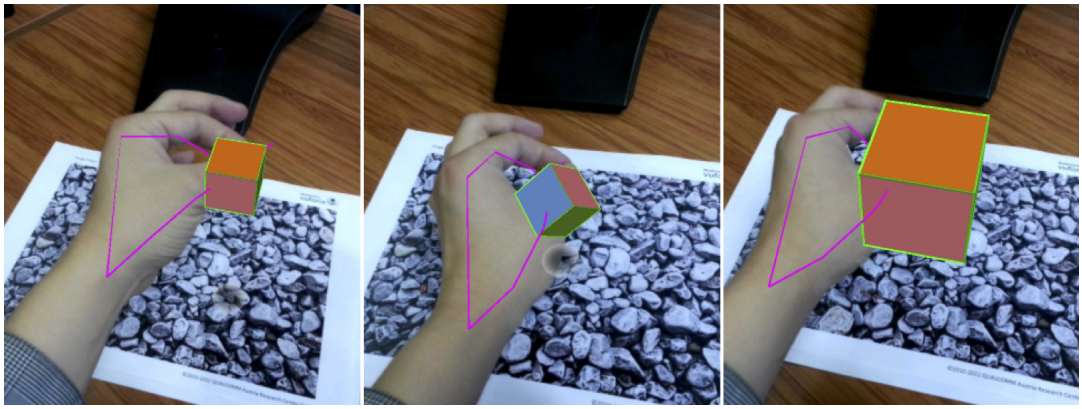

FIGURE 4.7: Examples of gesture-based canonical manipulations for our mobile AR application by participants.

In each trial, participants pressed the on-screen start button at the beginning of each task and pressed the stop button when they finished. The task performance was measured in terms of task completion time and placement error, and both were automatically measured and recorded by the system. The completion time was the time length recorded between the task start and the end, and the placement error was the difference of the position, orientation, or size of the operated object at the end compared with the excepted pre-defined values claimed in Section 4.3.1. At the end of each trial, participants answered a usability questionnaire. Once the participant finished all six trials, they were also asked to answer a post-experiment questionnaire to give more detailed subjective feedback. The whole user study took approximately 40 minutes on average.

4.3.3 Experiment Results

We recruited 18 participants (15 males and 3 females) whose ages range from 19 to 32 years old (M = 24.83, SD = 4.25) for the experiment. All of them except one had previous experience with using a touch-based interface, 12 had used hand gesture-based interface, and 11 had used AR interfaces before, and their AR experience were less about interaction but more focused on information display instead. Twelve participants answered right as their dominant hand, while 4 of them answered left, and the other 2 answered both. During the experiment, based on their preference, ten

participants held the device in their left hand and used the right hand for interaction, while for eight it was the other way around.

Task Performance

We measured the task completion time and placement error to compare the user's task performance when using touch and gesture-based interfaces for each experimental task. As both task completion time and placement error in some of the conditions were found to be not normally distributed (using the Shapiro-Wilk Test), we used non-parametric (Wilcoxon Signed-Rank with $\alpha = .050$) tests to compare the two conditions.

First we compare the task completion time between the two interfaces (Figure 4.8). The touch-based interface took significantly less time than the gesture-based interface for the translation task (Z = -2.548, $p = .011$; Touch M = 18.8 sec., SE = 1.71; Gesture M = 26.4 sec., SE = 3.362) and the scaling task (Z = -2.678, $p = .007$; Touch M = 10.7 sec., SE = 0.9; Gesture M = 15.2 sec., SE = 1.2).

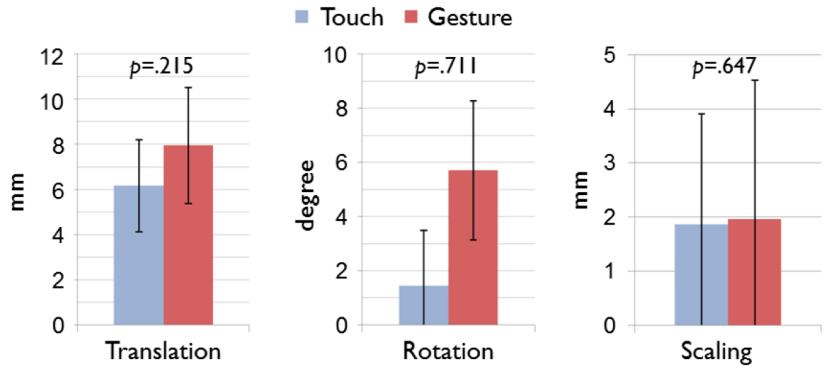

FIGURE 4.8: Task completion time (error bar: SE).

Although the gesture input normally just needs one spatial movement to complete these two tasks, the user had to move the hand in 3D space over a much larger distance and at a slower speed to keep the hand tracking working. In the touch screen condition, even though the user needed to touch several times with small scrolling motions on the screen to achieve three axis movements, they could do it quickly and over a short distance.

In contrast, the gesture-based interface (M = 25.9 sec., SE = 2.9) appeared to take around the same time as the touch interface on average (M = 30.3 sec., SE = 4.1) for the rotation task, and no significant difference was found (Z = -1.023, $p = .306$).

When comparing the placement error for each task, overall the gesture-based interface appeared to have more error on average, but no statistically significant

difference was found (translation $p = .215$; rotation $p = .711$; scale $p = .647$; see Figure 4.9). For the translation task, with the touch-based interface participants made average errors of 6.2 mm (*SE* = 2.0) while with the gesture-based interface the error was 7.9 mm on average (*SE* = 2.6). The touch-based interface had 1.4 degrees of error on average (*SE* = 0.3) in the rotation task, while the gesture-based interface had 5.7 degrees of error on average (*SE* = 4.5). For the scaling task, touch and gesture-based interfaces had 1.8 and 1.9 mm of errors on average (*SE* = 0.3 and 0.5), respectively.

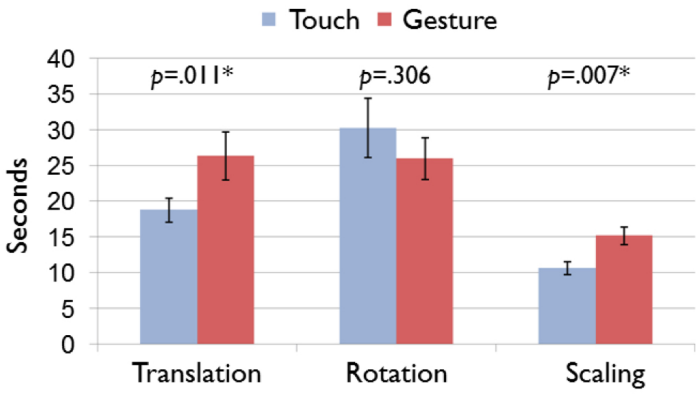

FIGURE 4.9: Placement error (error bar: SE).

One main factor affecting error was the accuracy of the hand pose tracking and estimation software. The average amount of the estimated joint and fingertip location error to the ground truth was around 5 mm, while the hand pose error was much bigger because it was calculated and estimated based on the joint angle.

Usability Questionnaire

After using each interface for each task, participants answered a questionnaire with eight subjective questions (see Table 4.1) about different aspects of usability of the interface. The questions were answered on a seven level Likert-scale rating, ranging from 1 (totally disagree) to 7 (totally agree).

We compared participants' answers between the two interfaces (touch and gesture) for each experimental task with the Wilcoxon Signed-Rank ($\alpha = 0.050$) test. Figure 4.10 presents three task results in a box plot.

For the translation task, both interfaces were rated above 6 on average for Q1, Q2, Q3 and Q6, reflecting that the participants felt they were performing well, the interface was easy to use, easy to learn, and useful to complete the tasks. Participants rated both interfaces having not much stress neither mentally (Q7. Touch(T): *M* = 1.944, Gesture(G): *M* = 2.167) nor physically (Q8. T: *M* = 1.833, G: *M* = 2.444). No statistically significant difference was found between the two interfaces in all of these

TABLE 4.1: Usability questions

	The given interface was:
Q1	Performing well
Q2	Easy to use
Q3	Easy to learn
Q4	Intuitive – you feel straightforward to learn and use the method without much thinking
Q5	Natural – you feel the method similar to your daily interaction behaviors
Q6	Useful for tasks
Q7	Mentally stressful
Q8	Physically stressful

six questions. However, for the questions asking how intuitive (Q4) and natural (Q5) was the interface, participants gave a significantly higher (Q4. Z = -2.263, $p = .024$; Q5. Z = -3.007, $p = .003$) rating to the gesture-based interface (Q4. M = 6.444, Q5. M = 6.389) rather than the touch-based interface (Q4. M = 5.444, Q5. M = 5.000).

The scaling task showed similar results as the translation task, except that only the naturalness (Q5) showed a statistically significant difference, which indicated that the gesture-based interface (M = 6.000) was rated significantly more natural (Z = -2.263, $p = .024$) than the touch-based interface (M = 5.056).

In terms of the rotation task, participants gave a higher rating to the gesture interface for the first six questions (Q1∼6), and there was a significant difference for four questions (Q2∼5). These results show that participants felt the gesture-based interface (Q2. M = 6.222; Q3. M = 6.389) was significantly easier to use (Q2. Z = -2.226, $p = .026$) and learn (Q3. Z = -1.996, $p = .046$) compared to the touch-based interface (Q2. M = 5.222; Q3. M = 6.667). Meanwhile, the gesture-based interface (Q4. M = 6.167; Q5. M = 5.667) was more intuitive (Q4. Z = -2.994, $p = .003$) and natural (Q5. Z = -3.044, $p = .002$) than the touch-based interface (Q4. M = 4.500; Q5. M = 4.222). No significant difference was found for Q6 and Q7 as most of the participants did not feel much stress for both interfaces (M = 2.556 or less for all).

Comparing the averaged ratings across the all tasks showed that the gesture-based interface was significantly more intuitive (Z = -3.162, $p = .002$) and natural (Z = -3.395, $p = .001$) compared to the touch-based interface, while both interfaces were well accepted by the participants with getting rated around 6 or above for performing well, being easy to use and learn, and useful, and yet not being very stressful to use. Figure 4.11 presents a box plot that summarizes this result with whiskers representing minimum and maximum values.

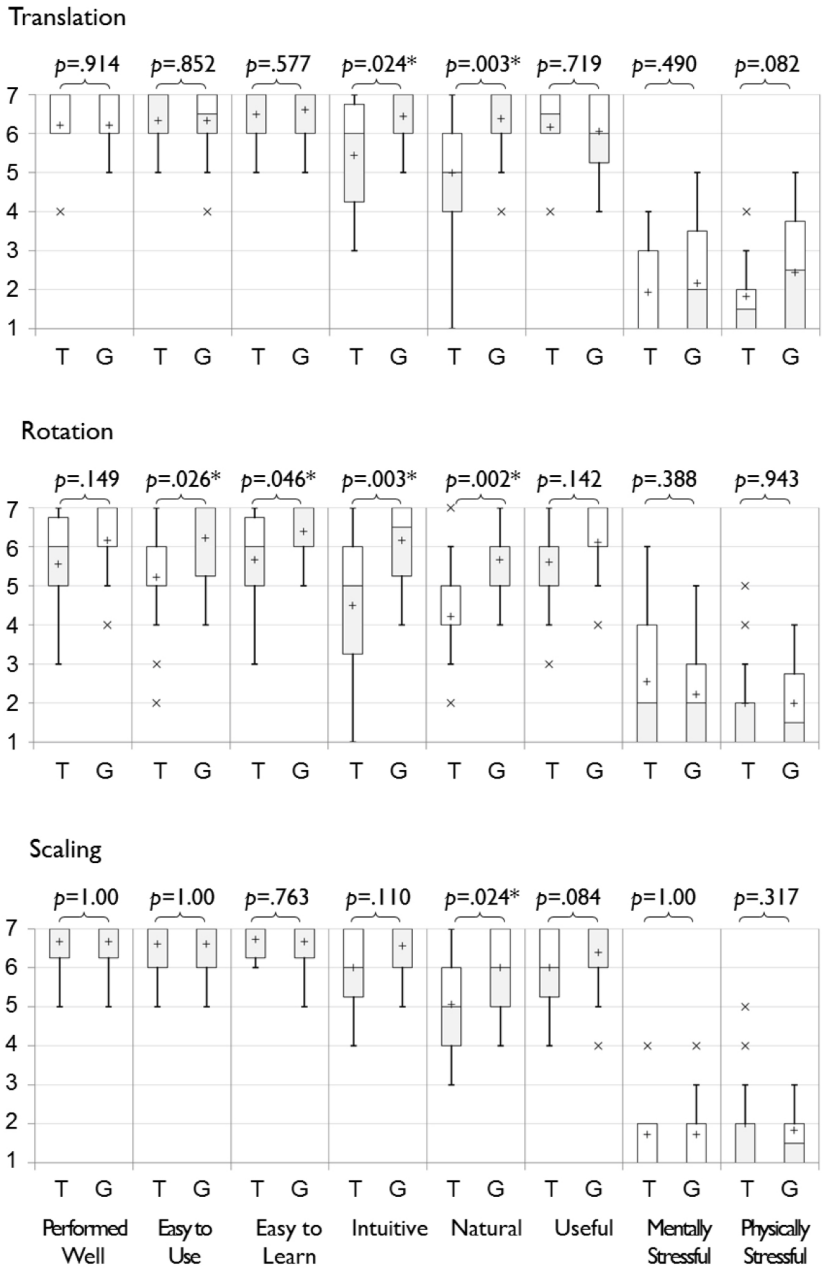

FIGURE 4.10: Results of usability question in Likert-scale with three type of tasks individually (T: touch, G: gesture; +: mean, ×: outlier with 1.5 IQR, *: statistically significant).

With good internal consistency (Cronbach's $\alpha = .858$), comparing the average of the ratings across all questions and tasks showed that overall the gesture-based interface (M = 6.220, SE = 0.126) received significantly higher (Z = -2.391, $p = .017$) ratings for the usability questions compared to the touch-based interface (M = 5.810, SE = 0.172).

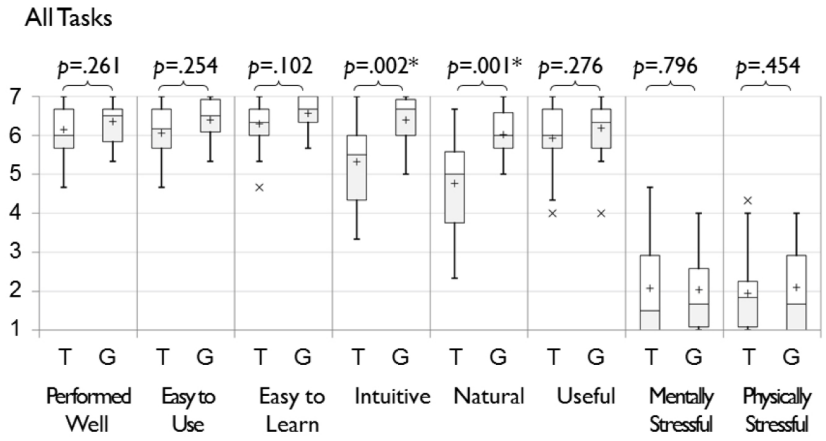

FIGURE 4.11: Results of usability question in Likert-scale with all type of tasks aggregated (T: touch, G: gesture; +: mean, ×: outlier with 1.5 IQR, *: statistically significant).

Post-experiment Questionnaire

At the end of the questionnaire, we also asked participants to choose an interface that they prefer to use for each experimental task and in general with the three tasks combined (see the full questionnaire in Appendix A). Figure 4.12 shows the result of user's preferences. The gesture-based interface was preferred by more than half of the participants (10 and 11) for doing the translation and rotation tasks, while half of the participants chose it for the scaling task. Only 4 participants chose the touch-based interface for the translation and scaling tasks, and 6 chose it for the rotation task, while the rest chose to use both. For general task completion, 8 participants preferred the gesture-based interface, while 7 preferred the touch-based interface, and the rest choosing both.

Based on the responses to the open questions that we asked at the end of the study, we can explain the user preferences above to some extent. First, 56% of people preferred using gesture alone compared to only 22% preferring touch alone for translation. Since the operational object and the input gesture are in the same 3D coordinate system, this makes gesture manipulation very natural and intuitive. In contrast, the touch-based translation is only easier for 2D movement and not convenient for specifying 3D displacement. For example, in our case, extra two finger touching is required for accessing the third axis for the touch-based interface.

A few users claimed that it could be better for the translation task if both touch and gesture-based interfaces were integrated together and available at the same time, so they can easily switch between these two methods. For example, gesture control could be used for large-scale placement while the touch-screen for adjusting the final

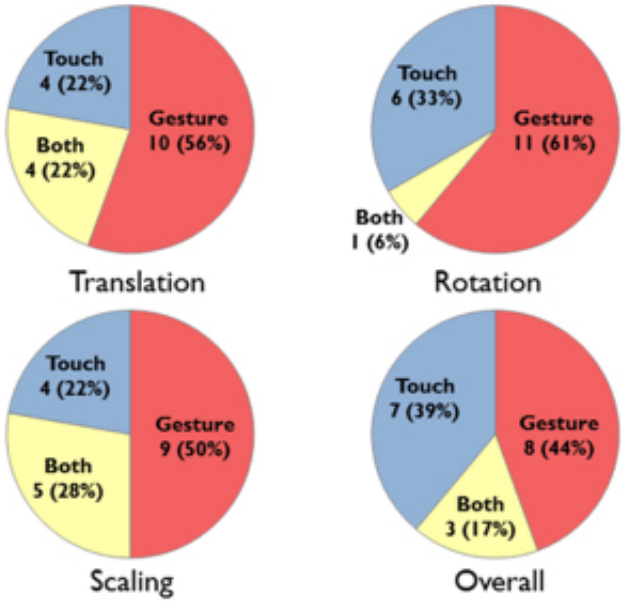

FIGURE 4.12: Users' preference.

position to achieve more precise results at the end. This combination could also be applied when the user needs to complete a time-consuming translation with a high accuracy requirement. The gesture input might be easily for users to use, but their hands and arms might suffer from fatigue when they have been doing the gesture interaction for a long time. If this is the case, they can change to use the touch interface instead to make it less physically stressful.

A total of 61% of participants preferred using gesture alone compared to only 33% preferring touch alone for rotation. The rotation design also required double finger touching to control the third axis rotation. Even only using a single finger touch but with two axes activated, which is commonly designed in the mobile applications when people make a 2D touch movement on the screen, the rotation can be very confusing for people. In contrast, rotation based on the hand pose from a single hand gesture is much more natural and easy to follow, while using less physical stress, compared with the separated multi-touching required for different axes.

Scaling is conducted with a similar simple scrolling gesture on the 2D screen and space movement in the 3D space for touch and gesture interaction respectively. A total of 50% of people preferred using the gesture alone compared to only 22% preferring touch alone for scaling. People felt that the touch scaling was easy to use but not so intuitive for them because they were already used to the two-finger pinching for resizing pictures on their personal mobile devices. However, in our case the method is slightly different as we only use one finger.

The overall preference is so much closer between touch and gesture. This is not

unexpected since the participants gave a high rating to both interfaces in terms of their performance, and they claimed that with both methods they could accomplish tasks efficiently, although each of them has their own advantages and drawbacks.

4.3.4 Discussion

Users felt that they needed to learn and practice to use the touch-based interface and sometimes instructions are needed because the interface does not mimic other interaction patterns in their life. One main shortcoming of touch-based interface is the limited screen size. Finger occlusion of on-screen content and shaking of screen touching problems can also reduce the final performance.

Although the user could feel physical stress by using both hands for holding and interacting with content using gesture-based interaction methods, they claimed that the gesture input was much more enjoyable to use with less learning and thinking. This may be because the gesture interfaces that we provided was designed to use the same gestures as used in real daily interactions, with having more straightforward operational steps with much fewer UI layers.

Currently one of the biggest drawbacks for both input techniques is difficulty of accurate depth perception. With the AR view on the handheld device it was sometimes difficult to know exactly the position of the virtual content in the real world. For example, some users believed that they finished the translation task just because both virtual cubes were overlapped with each other from their current perspective. However, when they checked the result from other viewing angles, they found that the cubes were not aligned at all. It took most people a few seconds to locate the 3D position of the target object even with the help of shadows and a 3D indicator. There was also an issue with depth occlusion. Since the virtual object is rendered on the top layer of the final AR view, the virtual content always appears on top of the user's hand, regardless of the depth of the hand. This means that some occlusion from hand that normally happened in the real-world was not displayed correctly, which made it more difficult for the users to understand the spatial relationship. This problem could be solved by adding accurate depth occlusion for the interacting hand.

4.4 Conclusion

In this study, we presented a novel client-server gesture framework that allows us to easily simulate and evaluate different free-hand gesture interaction methods on

mobile devices. Using this framework, we developed a finger-based and skeleton-based 3D gesture input method, with which users could manipulate AR objects for translation, rotation and scaling in 3D space with their bare hands in midair.

From the study results, we concluded that the 3D gesture-based input is as easy to learn and use as the traditional touch-based method. Although it is not very suitable for long periods of use because of the physical stress and needs to be improved in terms of task performance, 3D free-hand gesture interaction was considered statistically significantly more intuitive and natural to use, and got a higher rating than the touch-based interface in user preferences for the canonical interaction tasks.

While this framework is good for prototyping, in the future, we will need to develop a system running entirely on the smartphone. This will be possible when depth sensing becomes widely available on mobile devices. Using gestures in combination with other modalities of interfaces could also be an interesting direction for future research.

Chapter 5

Prototype Applications

In this chapter we describe the following three applications based on the client-server framework presented in Chapter 4; (1) advanced 3D gesture interaction for modeling and animation for mobile AR (Section 5.1), (2) a multi-modal interface for a wearable mobile AR system (Section 5.2), and (3) an augmented exhibition podium that supports natural free-hand 3D interaction (Section 5.3). The ability to use our gesture interaction framework for such different types of application show how flexible the framework is, and useful for prototyping many different gesture-based mobile AR interfaces.

5.1 Advanced 3D Gesture-based Interaction

Previous hand gesture-based interfaces for mobile AR (Section 4.1 & 4.2) have been mainly studied for canonical manipulations, such as selection, translation, rotation, or scaling. However, using hand gestures for more complicated AR modeling tasks, like shape editing or animation path design, has not been well explored.

In this section, we study hand gesture-based interfaces for real-time 3D AR modeling and animation on handheld mobile devices. Based on our previous client-server framework for gesture interaction, we present four manipulation methods with an emphasis on the most common modeling tasks in an AR environment. With this AR application, the user can create a virtual object, change its shape by dragging its faces or vertexes, and animate its spatial movement with a playback function. Further practical scenarios could benefit from this technology like rapid prototyping of 3D designs. Designers could quickly create virtual objects with their bare hands on their mobile phone and then refine the details on a desktop computer using CAD software.

5.1.1 Interface Design

The "*World Builder*" concept video[1] inspired us through a very detailed demonstration of how natural hand gestures can be intuitively used for 3D virtual object modeling, texturing, and animation. In this film, a man uses both his hands as tools to build a virtual town for the woman he loves; his fingers become tools of power able to create with exquisite delicacy – he can build, sculpt, paint, edit all the virtual content easily with various natural gesture interfaces.

In our project, we try to bring gesture-based interaction methods to the one-handed mobile AR system described in Section 4.2. We designed our gesture interface to include four new manipulation techniques: object creation, face extrude and subtraction, vertex dragging, and animation path creation and playback. Three basic hand gestures are involved; (1) one finger virtual touch for face and vertex selection, (2) two finger pinching for object creation, face and vertex dragging, and (3) two finger pinching for animation path drawing.

To simplify the interaction design, we split the input types by activating different editing modes via buttons on the touch screen; we also only consider one single basic geometry shape, a cube, testing the current prototype. Although the same gesture has different metaphors under different modes, only pre-defined gestures can be recognised and perform certain functionalities once the specific mode is activated.

Object Creation

We use a single pinch point as the spawn point of the new virtual cube and if the user moves the pinch gesture in the mid-air, the cube will also grow synchronously to show the generated size. When the user releases the pinch gesture, then the cube creation will be completed (see Figure 5.2a). Note that this is different from a bimanual gesture-based method, with which the user makes the pinch gesture with both hands as the diagonal of a new cube.

Face Extrude and Subtraction

Changing the shape based on faces is another important modeling technique. The user can virtually touch the desired face of the cube and drag the selected face with the pinch gesture (see Figure 5.2b). Three steps are used to complete this interaction. First, the user uses their index finger to virtually "*touch*" the target faces in 3D space. Next, the selected face will be highlighted blue (Figure 5.2b). Finally, the user can then move the pinch gesture towards or away from the cube; if the pinch gesture is away from the cube, the selected face will turn green indicating extrusion and the face will grow towards the pinch gesture; if towards the cube, the selected face will

[1] http://www.youtube.com/watch?v=QP3YywgRx5A

turn red indicating subtraction and the selected face will shrink towards the pinch gestures. Releasing the pinch gesture will end the face manipulation.

Vertex Dragging

To change the shape of the 3D model with more detailed control, vertex manipulation is necessary, which shares a similar procedure as the face editing. The user can use their thumb and index finger to approach the target vertex, and the system will automatically attach the closest vertex located inside a threshold range (less than 10 mm) among the near targets. This will help user to easily select the vertex rather than searching for a tiny point in the 3D space without clear depth perception. Then the user can drag the selected vertex in 3D space and the object shape will be changed in real time (see for example Figure 5.1).

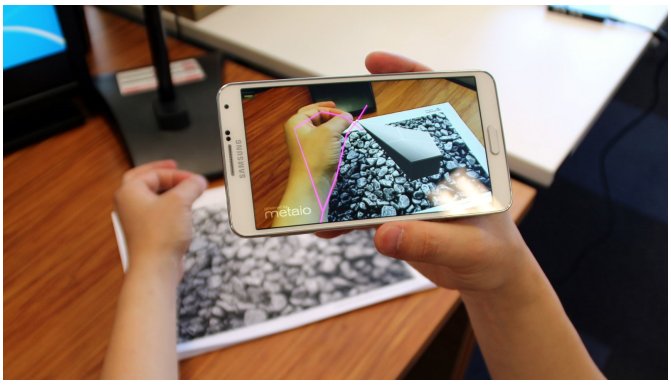

FIGURE 5.1: The user is editing with a virtual cube by dragging a selected vertex using freehand input on the mobile AR device. A virtual hand skeleton is textured on the real hand image, and an augmented cube is appearing overlaid on the tracked image.

Animation Path Creation and Playback

After the 3D model editing is completed, basic animation can also be designed based on the pinch gesture movement path. Once the animation mode is activated, the user can begin creating the animation path by using the pinch gesture - the movement of a pinch point in 3D space (indicated by a small white sphere in Figure 5.2c) will be recorded as a new path automatically and displayed by dropping green animation path points (Figure 5.2c). Once the path is successfully generated, the user can release the pinch gesture to active the playback feature of the created animation path; this will cause the cube to automatically move by following the recorded gesture path from the start point to the end point.

5.1.2 Discussion

A sample application supporting all four functions above was developed, and was demonstrated to 5 university students for preliminary evaluation. The students were

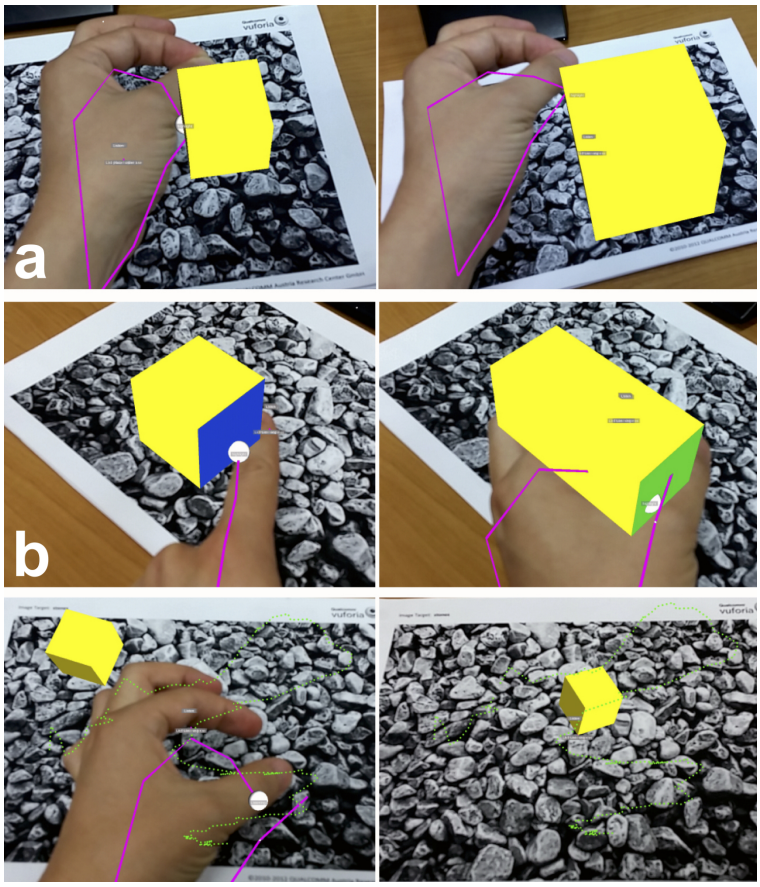

FIGURE 5.2: 3D advanced manipulations with free-hand gestures for mobile AR; a) object creation, b) face extrude and subtraction, c) animation path creation, animation playback.

asked to try all four interaction methods described in Section 5.1.1 until they fully understood the operation procedures and completed the sample tasks as shown in Figure 5.2 and Figure 5.1.

Although no subjective evaluation was performed, user feedback was mostly positive. The participants claimed that the interaction methods in the four cases were natural and intuitive. They can easily learn and practice to use the hand gesture-based 3D interaction methods since all interaction patterns mimic daily life behaviours. They felt the hand tracking was responsive and the image tracking was robust enough. Despite this, the application dropped the frame rate a few times during the whole procedure because of the network delay.

In term of modeling, we have only tested a virtual cube in our prototype as the initial modeling object. Although lots of 3D model designs are based on basic geometry, such as sphere or cube, other components are added to complete the design. Importing and editing of external models could also be supported in the future for further editing in real-time AR environment.

Currently, the face and vertex selection and editing are sufficient for simple

object modeling, but fail to provide the other necessary functionality for editing and manipulating an object mesh into a more complicated 3D model. Another desired feature for modifying the AR scene would be texturing techniques using 3D gesture input.

In terms of animation, we detect the pinch gesture and apply only the spatial position of the pinch point as the movement path for virtual object without pose or orientation info. In the future, we will combine the palm pose data into the animation path design, adding orientation for the movement playback.

Mid-air interaction can affect the precision and the quality of interaction caused by the lack of stability. Due to this, the initial design of thumb to middle pinch for mode selection was replaced by touch input due to the similarity between other pinch gestures. This issue could be overcome by improving the gesture detection algorithm. Physical stress is another serious but expected issue for gesture interaction on the handheld device. Although the mobile phone is relatively light compared with most tablets, both the user's hands and arms might suffer from fatigue while they are doing gesture interaction from holding the device. We would like to investigate the interfaces combining gesture with other interaction modalities, like speech recognition and eye tracking, to improve the efficiency on handheld and wearable AR devices. For the mode selection and playback control, we could also integrate the touch-screen input with the gesture input to simplify the system design. However, this can be a distraction for users since they have to switch their input methods between stretching the hand behind the mobile camera and taking it back to the screen surface. Other natural interaction methods like speech input could be more appropriate to overcome the distraction issue while keeping the intuitiveness of the AR interaction.

Participants enjoyed being able to edit the AR object directly on their mobile phones with their bare hands instead of only viewing it from different perspectives by moving the phone itself without any further manipulations, or having to do these types of advanced editing on their PC separately. Many users expressed an interest in being able to experience the future self-contained version.

5.2 Multi-modal Interaction

A recent development in wearable computing is lightweight, head-mounted devices that incorporate sensing and display technologies, but their input methods have substantial limitations when compared to more traditional computing platforms. For example, Google Glass supports touch pad, button or speech input. In some

applications, there is a need for richer interaction, particularly those that involve viewing and manipulating 3D graphics in mobile AR.

As mentioned in Section 2.3, there have been a few earlier systems that explored gesture input with wearable computing, but mostly without a full 3D hand model. Meanwhile, there has been significant research on interaction methods for both hand and touch gesture based interaction, but less work has been done on wearable AR systems combining both hand and touch gestures.

In this section we describe a wearable AR system that we have developed using a multi-modal interface based on our previous framework (Section 4.2), and we present preliminary results from a pilot study. The main novelty of our work is that we have developed a prototype for incorporating rich 3D free-hand gestures into a wearable AR system, and explored interaction for the system by combining gestures with precise fine-scaled touchpad input. In the next section, we describe our design and prototype with these capabilities.

5.2.1 3D Hand Gestures for Wearable AR

We wanted to study full 3D hand tracking input for AR applications on Google Glass. However, Google Glass has limited processing power and only a single camera so it is difficult to perform 3D gesture tracking on the device itself. To prototype future gesture input capabilities, we combine the Google Glass with our earlier client-server framework (Figure 5.3). Although Google Glass has a see-through screen, it is only feasible to support video-based AR applications instead of see-through ones because of the small field of view. Figure 5.4 shows an augmented hand skeleton and teapot appearing overlaid on the tracked image on the Google Glass. In this case, we directly ported the implementation of the system in Section 4.2 from the mobile phone to the Google Glass with just a few updates on the UI design.

The system still has the same limitation that we mentioned before, which is that the user needs to keep their hands within the Three Gear interaction volume, preventing them from moving outside the desktop space.

5.2.2 Combining Hand Gestures with Touch Input

Our prototype system can support bimanual hand gestures, which is similar to previous work in multi-touch interfaces [98]. We are also interested in using hand gestures in combination with different input methods commonly available on wearable computers, such as the touchpad, for AR applications. This approach is in-line with work

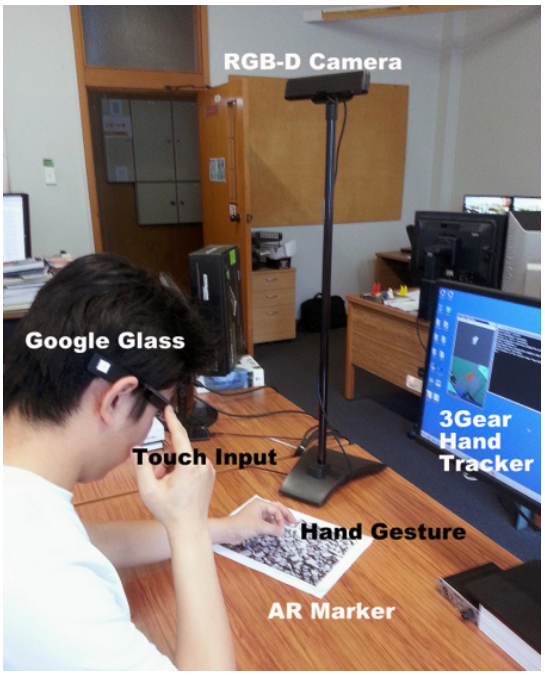

FIGURE 5.3: System setup: 3D hand gestures for wearable AR.

in cooperative bimanual interfaces [43] where users use different tools with each hand cooperatively.

Google Glass has an integrated high-resolution touchpad (1366 by 187 pixel resolution), so we used this to explore the combination of hand gesture and touch input in wearable AR applications. The goal was to explore how fast, but low resolution free-hand gesture input could be combined with touch pad input. There are several ways these technologies could be integrated together, and these can be mainly categorized into two cases, (1) asymmetric scale and (2) asymmetric task.

In the case of asymmetric scale, both hand gesture and touch input provide the same function, but with different scales. For example, users can use hand gesture for roughly placing an object, and use touch input to fine tune its position. With this combination, we can have intuitive and fast manipulation with hand gesture, while improving the accuracy by using precise and high-resolution touch input.

The other type of approach for combining two input methods is assigning different tasks or functions. In this case, hand gestures and touch are used to control different aspects of a single object or even control different objects independently. For instance, a hand gesture could be used to draw a 3D line, while the touch pad input could be used to change the thickness or colour of the line. Another example could be controlling different objects; hand gestures for manipulating objects, and touch input for controlling the view zooming.

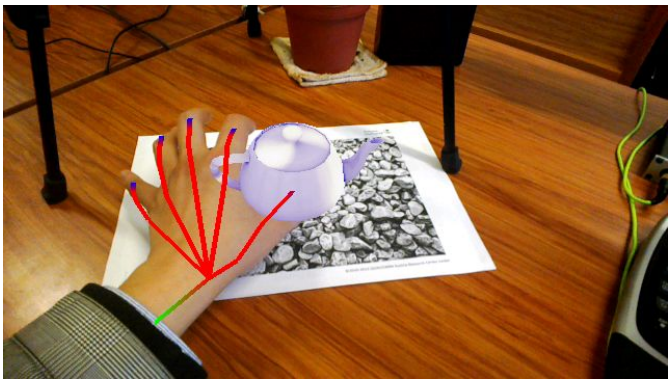

FIGURE 5.4: An AR scene displayed on a Google Glass screen with a user's hand tracked and overlaid with full-hand skeleton. The user can manipulate a virtual teapot in this AR scene by using pre-defined hand gestures.

5.2.3 Prototype Application

We are interested in using wearable AR systems for 3D scene assembly, so the first simple test application we developed was for fast and precise virtual object placing in 3D space. Wearing Glass, users will see a virtual object shown on the Vuforia trackable image, and may use a combination of free-hand gesture and touch pad input to move the object to a target position.

Figure 5.5 shows the screenshot of this application on Google Glass. With this application, the user can use one hand to point at a virtual teapot and a pinching gesture to select it, and then keep pinching while moving the hand to translate it, and eventually open the hand to release the control. The user can also use the other hand to scroll through the touch pad to change the teapot location. To apply the displacement on different axis, the user can make a single tap on the scroll pad to change activated axis, while the screen will display relative text for notification once the tap action is detected.

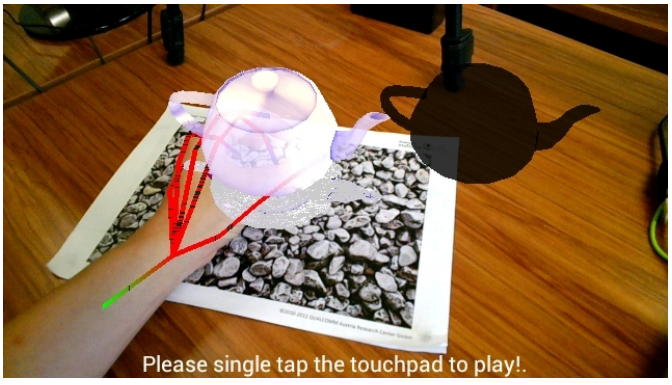

FIGURE 5.5: The user can pinch the purple teapot and move it to a target position indicated by a black teapot in space.

5.2.4 User Evaluation

To explore the usability of our prototype system we conducted a pilot user study. Five participants (all male) were recruited for the experiment. Their ages ranged from 22 to 38 years old (M = 30.00, SD = 5.958). Most of them had previous experience with using AR interfaces, and Google Glass, but had very limited 3D gesture knowledge.

Each participant was asked to select and move a virtual teapot in an AR environment to the target position shown as another teapot in different colour, as shown in Figure 5.5. Participants performed the task using three different interfaces: touchpad input only, hand gesture only, and a combined method with both hand gesture and touch input. The touchpad only and gesture only input methods were design in the same way as the interaction explained in Section 4.3, with replacing the touch screen with the touchpad instead. Participants were told to perform the task as fast and as accurately as possible, and the order of the conditions was randomized.

We measured user performance in terms of task completion time and position error based on the system log. We also collected subjective feedback with questionnaires answered in a Likert-scale (1: strongly disagree, 7: strongly agree). Table 5.1 summarizes the questions asked.

TABLE 5.1: Friedman test result

Question	p
The given interface was:	
Q1. Easy to use	0.092
Q2. Easy to learn	0.165
Q3. Intuitive	**0.029**
Q4. Natural	**0.018**
Q5. I was performing well	0.589
Q6. Useful to complete the task	0.232
Q7. Mentally stressful	**0.047**
Q8. Physically stressful	0.150

Using a one-way repeated measures ANOVA ($\alpha = .050$) to analyse the performance time results (Figure 5.6), we found a statistically significant difference among three interaction methods in terms of overall completion time (F (2, 8) = 5.416, $p = .033$). The further pairwise comparisons indicate that there was only a significant difference in operation time between the touch only input and hand gesture only input. Participants took more than triple the amount of time to complete the spatial translation with the touch based interface (M = 68.870 sec., SD = 36.445, SE = 16.298) compared to the hand gesture method (M = 23.216 sec., SD = 11.301, SE = 5.054). The participants using the combined interface spent less time than the touch only input, but more time than the hand gesture only condition (M = 49.566 sec., SD = 18.590, SE

= 3.649). Position errors showed no significant difference among three interfaces (F (2, 8) = 0.207, $p = .818$).

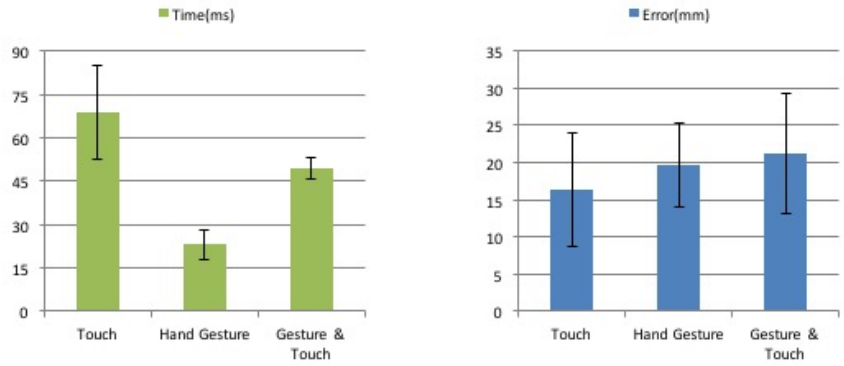

FIGURE 5.6: User performance in terms of task completion time (left) and position error (right).

Figure 5.7 shows the average survey results for each of the subjective questions. Analysing the results with a Friedman test, three of the questions showed statistically significant differences between the conditions (see Table 5.1).

Overall, the hand gesture obtained the most positive rating in most of the questions, while the combined method followed next except for mentally stressful. It is noticeable that the hand gesture was considered most intuitive and natural while involved the least mental stress. Touch input was least preferred by the participants, and was considered as the most physically stressful method, since it required lots of finger movement, especially doing the 3D movement using one axis after another. In contrast, with hand gestures, participants simply moved their hands to the point where they want to move.

While the combined method was rated more positive compared to touch input in most of the questions, participants found it was most mentally stressful. Participants said that they felt distracted when trying to coordinate their two hands using different interfaces, requiring more time to think about which interface they should use during the task. This should be considered for future work for designing interactions combining hand gesture and touch input.

5.3 Augmented Exhibition Podium

In this section we present an augmented exhibition podium that supports 3D free-hand interaction for visitors using their own mobile phones or smart glasses. Visitors can point the camera of their devices at the podium to see AR content overlaid on a physical exhibit from their device display, and at the same time use their bare hands

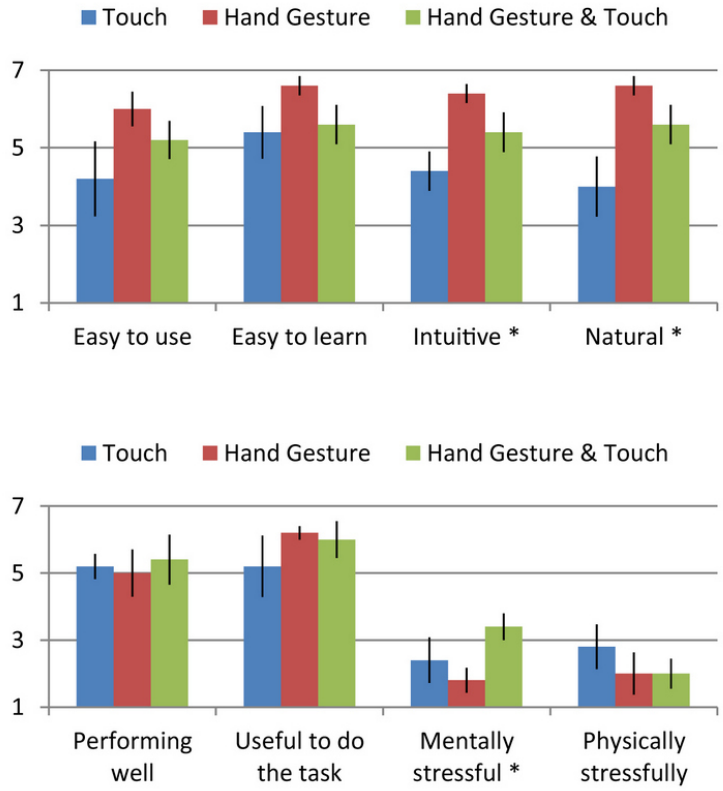

FIGURE 5.7: sability questions and results: Easy to use, Easy to learn, Intuitive, Natural, Performing well, Useful to do the task, Mentally stressful, Physically stressful (Error bar: +/- *SE*). The terms with significant differences are indicated by using an asterisk.

to interact with the AR content. For instance, they can use pinching gestures to select different parts of the exhibit with their fingers to view augmented text descriptions, instead of touching the phone screen.

Since we would like visitors to use their personal mobile or wearable device, it is not feasible to ask them to modify their devices to attach additional sensors. Instead, we propose to use environmental sensors built into the exhibition podium, and then send the tracking data wirelessly to the visitor's device. More specifically, our prototype combines vision-based image tracking and free-hand gesture detection via our client-server framework (Section 4.2), which enables users to use their hands with the augmented exhibition without requiring special hardware (e.g. a depth sensor) on their personal devices.

We report on user feedback from a pilot user study, and discuss the strengths and weaknesses of the prototype augmented podium. Results from our pilot user study shows that the prototype system is as intuitive to use as a traditional touch-based interface, and provides a more fun and engaging exhibition experience.

5.3.1 System and Interface Design

System Design

Figure 5.8 shows the design of our augmented exhibition podium. There can be many exhibition podiums placed separately in a museum, all of which are connecting to the same wireless network with their own unique network serial number. When users visit the exhibition with a personal device (e.g. smartphone, tablet, or smart glasses), they are provided with a link (e.g. QR code, NFC tag) to download and install the AR application from. Thus, the user can run the application to scan the QR code attached on the podium, which will automatically wirelessly connect the device to the identified exhibition podium server.

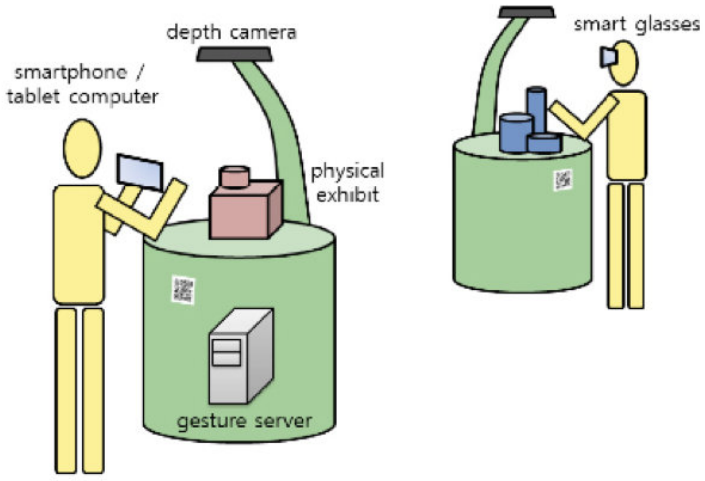

FIGURE 5.8: Concept design of augmented exhibition podiums with free-hand gesture interfaces.

In our system, once the connection is established, the AR visualization with vision-based tracking begins on the user's mobile device, and a computer inside the podium (i.e. the gesture server) also starts to track the user's hands with a depth camera mounted above the podium. The mobile application communicates with this gesture server to receive hand-tracking information and applies this gesture input into the AR scene. The system only supports two hands at this moment, however, other users can connect with the podium server to observe the interaction and the content.

With the above configuration, the gesture interaction is decoupled from the AR tracking, so the system can support rich free-hand gesture interaction without requiring additional sensors on the users' mobile device. Since the hand tracking is

processed on a separate computer, the mobile device only needs to generate the AR visualization. This requires less computing resources, so a variety of low-end mobile devices can be used in this system. Another benefit of this design is that users can interact with the exhibit even when their hands are out of the view of their mobile device camera. In contrast, if the sensors were mounted on the mobile device, then the user's hands would have to be within the camera view at all times.

Interface Design

In an exhibition, visitors usually check the label to learn interesting information about an exhibited piece. To extend the exhibition experience for users via mobile AR techniques, we developed free-hand gesture-based interfaces for visitors to activate the exhibit's augmented information. One finger pointing (Figure 5.9) was tested but discarded because of the lower stability and accuracy compared with two finger pinching, which was chosen for our exhibition interaction (Figure 5.10a).

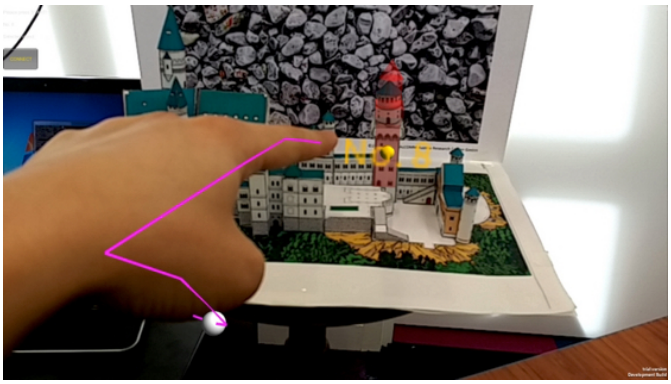

FIGURE 5.9: A square tower is highlighted in red when the user is pointing to it with the index finger.

5.3.2 Implementation and Performance

We set up the exhibition podium as shown in Figure 5.10b, being similar to the configuration of Section 4.2.1. A PrimeSense sensor was positioned 80 cm above the interaction space, facing downwards, and an object placed on the podium right below the camera. A printed image background was placed under the physical exhibit and used as an AR trackable marker. The whole software implementation was extended from our previous client-server framework.

We ran the server software on the same Dell laptop used in Section 4.2.2 and the client software on a Samsung Galaxy Note3 smartphone (ARM 2.3 GHz Quad-Core Chipset, 3 GB RAM, Android 4.4 OS) or an Epson BT-200 head mounted display (TI OMAP 1.2 GHz Dual-Core Chipset, 1GB RAM, Android 4.0 OS). The hand tracking runs at 25 FPS on the laptop with a fingertip accuracy of around 5 mm, and the AR

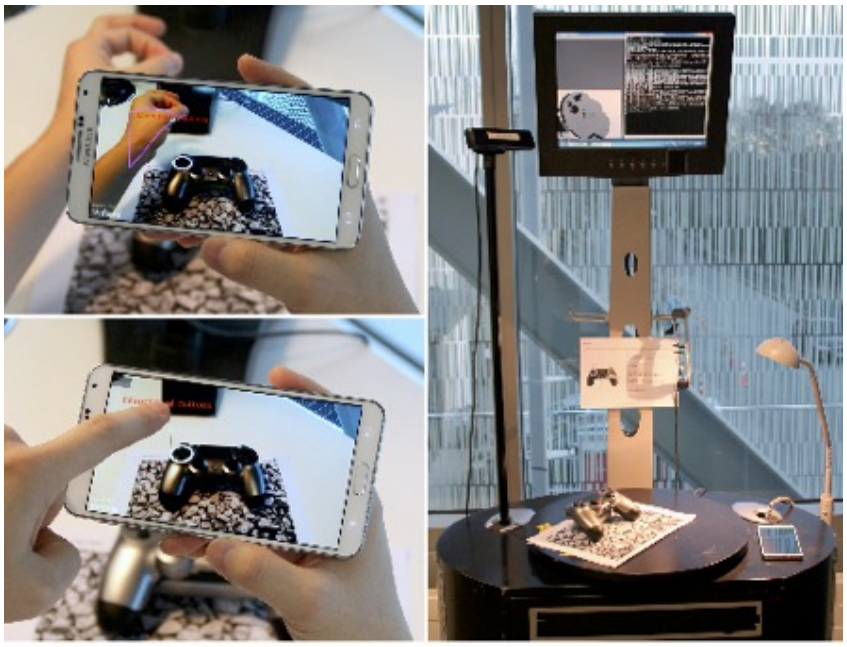

FIGURE 5.10: Gesture-based interaction and Touch-based interaction for AR Exhibition (left); Augmented Exhibition Podium Setup (right).

tracking runs at around 30 FPS with 1280 by 720 pixel camera resolution on the Note3 and at 20 FPS with 640 by 480 pixel camera resolution on the BT-200. While there are smart glasses equipped with optical see-through display, we noted that optical see-through displays have less accurate registration, need to be calibrated, and are not widely supported in mobile or wearable devices. Thus we limited our prototype system to only work in a video see-through mode for all devices.

5.3.3 User Evaluation

To explore the usability of our prototype system we conducted a pilot user study. We used the augmented exhibition podium and the interaction method described in Section 5.3.1, and compared our proposed gesture-based interface in a basic AR exhibition scenario to a touch-based input method described below.

Experimental Procedure

The experiment was a within-subject design with the type of interface as an independent variable. The experimental task was to access the augmented description of a Sony PlayStation Dualshock 4 controller, one of latest game console input devices, by using two different interaction methods. Each participant was asked to activate the augmented text by selecting virtual spheres that are linked to the target part of the exhibit in 3D space, which is similar to the traditional instruction menu. Participants performed the task using two different interfaces: (1) screen touching and (2)

gesture pinching (Figure 5.10). Participants were told to use each interface to access exhibit information as much as possible for 5 minutes. The order of the conditions was counter balanced to avoid carryover effects. After finishing both conditions, participants answered a questionnaire that asked them to choose which interface they preferred in an AR exhibition, and to write down the benefits and problems of each interface.

Twelve participants (8 males, 4 females) were recruited for the experiment. Their ages ranged from 23 to 34 years old (M = 27.8, SD = 4.2). Ten of them had previous experience with using AR interfaces, and nine of them had limited 3D gesture experience, mainly from Kinect gaming experiences. All the participants had used touch interfaces, with 11 participants stating that they used them frequently. However, none of them had knowledge about the testing exhibit.

We collected subjective feedback using questions answered on a Likert-scale (1: strongly disagree, 7: strongly agree). Table 5.2 summarizes the questions asked.

TABLE 5.2: Usability questions

	The given interface was:
Q2	Easy to use
Q3	Easy to learn how to use
Q4	Intuitive – you feel straightforward to learn and use the method without much thinking
Q7	Not mentally stressful
Q8	Not physically stressful
Q6	Useful to access the exhibit information
Q7	Fun to interact with the exhibit
Q8	Engaging to interact with the exhibit

Results and Discussion

Figure 5.11 summarizes the results of the usability questions analysed by using a Wilcoxon Signed-Rank test ($\alpha = .050$). Q3 is the only usability item that did not show a statistically significant difference between the conditions (Touch: M = 5.750, SD = 1.115; Gesture: M = 5.0833, SD = 0.996; Z = -1.381, $p = .167$). Participants claimed that although the free-hand gesture input was the most natural to use, they have gotten used to touch input by frequent use in their daily life and believe that the touching could be intuitive for them as well.

In contrast, statistically significant differences were found between the two conditions on Q1 (Z = -2.871, $p = .004$), Q2 (Z = -2.588, $p = .010$), Q4 (Z = -2.555, $p = .011$), Q5 (Z = -2.956, $p = .003$), Q6 (Z = -2.124, $p = .034$). The mean value of all these items show that the touch interface was rated significantly higher than the gesture

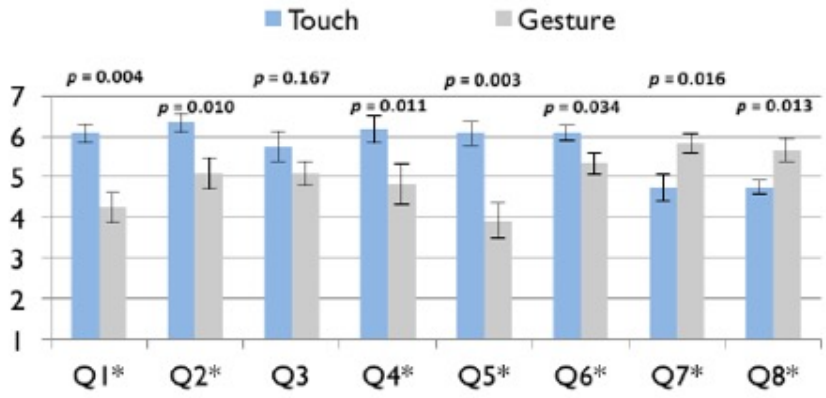

FIGURE 5.11: Usability questions and results with detailed values (Error bar: +/- SE). The items with significant differences are indicated by using an asterisk.

interface, and was considered easier to use (Q1) and learn (Q2), and was not as mentally (Q4) and physically stressful (Q5), and was more useful for accessing the exhibit augmented information (Q6).

Based on the comments from the participants, depth perception was identified as the main challenge for gesture interaction. Although the system provided an overlaid virtual skeleton and a sphere cursor with occlusion effect to help users to identify the spatial location in the augmented scene, the participants still needed more time to move their pinching fingers to the target area correctly. Participants claimed that their hands and arms might suffer from fatigue in this case, and this could also lead to the mental frustration as well. With all of these negative influences, they felt that the gesture method was not so useful for accessing the exhibit information as touch input.

In terms of Q7 ("Fun to interact with the exhibit") (Z = -2.412, $p = .016$) and Q8 ("Engaging to interact with the exhibit") (Z = -2.495, $p = .013$), statistically significant differences were also found between two conditions, but the gesture-based method got a significantly higher rating than touch input.

The overall preference is close between the gesture and touch: 7 participants chose the touch-based interface and 5 chose the gesture-based one. Participants believed that the gesture input has a huge potential for AR exhibits once the gesture tracking becomes mature enough, because of its ability to bring more fun and engagement into the exhibition experience.

5.4 Conclusion

In this chapter, we developed three applications based on our proposed client-server framework to demonstrate the value of our 3D gesture-based interface.

We first explored the possibility of the gesture interface for real-time 3D AR modeling and animation detailed with four manipulation methods on handheld devices. The simple evaluation confirmed that with a little training, participants can indeed perform basic 3D modeling with their hands in a mobile AR environment, and more rich and complicated AR interaction with mobile AR configuration can be achieved in the future. We then investigated the usefulness of combining 3D gesture interaction with touch input on wearable AR computers. The main novelty of this work is that both hand gesture and touch input provide the same function, but with different scale. The user can choose hand gesture for roughly placing an object, and change to touch input to fine tune its position. From the pilot study, we found that this system has intuitive and fast manipulation with hand gesture, while improving the accuracy by using with precise and high resolution touch input. This is just a work in progress and we hope to improve our implementation by using a small depth camera that is completely wearable. In the last section, we described an augmented exhibition podium that allows visitors to use free-hand gestures for interacting with exhibits in AR mode with various mobile and wearable devices, including smartphones, tablets, and smart glasses. Using their own mobile devices, visitors can use their bare hands to pinch above different parts of the exhibit to view related information. The study results indicate that free-hand gesture interaction techniques are fun and engaging to use for an augmented exhibition podium. We plan to extend the exhibition system by supporting collaboration between multiple users around the same podium.

With these applications, we deployed our 3D gesture-based canonical manipulation studied in Chapter 4 to more complex and realistic interaction tasks, and there are some unique difficulties that arise and need to be addressed later. For example, to complete more advanced gesture command, it is more suitable to give combined manipulation instead of to decouple it since this will reduce the whole operation steps. However, this will also lead to much more unpredicted input commands because of the immaturity of the gesture detection in real scenarios, which will make the gesture input unusable in some ways. Beside, users felt distracted when trying to coordinate their two hands using the multi-modal interface, which has a more steep learning curve and requiring more time to organize the right gesture operations.

Chapter 6

Conclusion

This goal of this thesis was to investigate the usability of free-hand gesture-based natural interaction in the context of mobile AR systems running on a handheld mobile phone or head-worn computer. In particular, our focus was on exploring how natural gesture-based interaction can be effectively integrated into mobile AR systems, and if it can be used to enhance a mobile AR system's usability in canonical AR manipulation tasks, providing more natural and intuitive user experience than touch-based interfaces while being as easy to learn and use.

To conduct this research, we developed two types of markerless gesture-based input technologies (self-contained design and client-server framework) with four different handheld mobile AR implementations, and one type of multi-modal interface with one wearable mobile AR implementation. We evaluated these implementations in four user studies with mobile AR systems. We first focused on comparing a 2D markerless fingertip-based gesture input technique with a freeze view touch method and a common free view touch-based interface for virtual object manipulation (Section 3.2). The next two experiments evaluated a 3D markerless fingertip-based interaction using an RGB-Depth camera attached to the handheld device, and a markerless skeleton-based 3D input method within a client-server framework, comparing both with a traditional touch approach across three fundamental scenarios with varying tasks (Section 3.4 & 4.3). We further explored a wearable multi-modal interface incorporating rich 3D free-hand gestures and touchpad input for spatial translation tasks (Section 5.2). We studied these four interfaces separately and presented our results in the corresponding chapters.

In this final part, we give an overview of all our results from a global point of view, grouping together all the lessons learned and forming a concise set of general considerations. These can be useful as high-level guidelines for interface designers who want to integrate free-hand gesture into mobile AR systems. Finally, we conclude the chapter highlighting promising directions for future research on the topic.

6.1 Lesson Learned

The research hypothesis of our research is that the key advantage of natural gesture in mobile AR systems is the intuitive and natural interaction metaphor that it provides. First of all, our experimental results confirm and strengthen this hypothesis (Q1).

- The 2D markerless gesture-based interaction for mobile AR did not show significant differences compared with the freeze view touch and the free view touch in terms of performance confidence, ease of learning the interaction method, and naturalness. This gives an indication that the users felt that the 2D gesture-based interaction is as easy to learn as traditional methods, and it was thought to be as natural as the other two touch-based interaction methods, while providing a more interesting and engaging mobile AR experience.

- The 3D markerless gesture-based interaction was considered statistically significantly more intuitive and natural to use, and got a higher rating than the touch-based interface in user preferences for the canonical interaction tasks that it was tested with. Although users felt more physical stress by using both hands for holding and interacting with content in the gesture-based interaction methods, they claimed that the gesture input is much more enjoyable to use. Users felt that they can use the gesture-based interface with less instruction compared with a touch-based interface for 3D mobile AR tasks, since the gesture interface mimic the familiar interaction patterns in their daily life.

- Multi-modal interaction of the 3D markerless gesture-based interaction combined with touch-based input on a head-worn wearable device enhanced the 3D markerless gesture-based method, and obtained the most positive rating in most of usability questions, except for how mentally stressful it was. It is noticeable that the hand gesture was considered most intuitive and natural and users said that they felt the lowest mental stress in this condition. Touch input was least preferred by the participants, and was considered as the most physically stressful method.

However, free-hand gesture-based input has certain limitations for mobile AR interaction that could be improved in the future (Q2).

- The gesture-based interaction appeared to have worse performance compared to the touch-based interaction methods. This seems to be mainly due to the immatureness of our gesture recognition software, as well as the extra time

consumed on dwell time selection techniques. To deal with these issues and improve the usability of the gesture interface, a steadier fingertip or hand skeleton detection algorithm should be developed in the future.

- One of the biggest drawbacks for 3D input techniques is difficulty of accurate depth perception. With the AR view on the mobile device it was sometimes difficult to know exactly the position of the virtual content in the real world. It took most users a few seconds to locate the 3D position of the target object even with the help of shadows and a 3D indicator. There was also an issue with depth occlusion. Since the virtual object is rendered on the top layer of the final AR view, the virtual content always appears on top of the user's hand, regardless of the depth of the hand. This means that hand occlusion that normally happens in the real-world was not displayed correctly, which made it more difficult for the users to understand the spatial relationship between the hand and virtual objects. This problem could be solved by adding accurate depth occlusion for the interacting hand.

- Users felt that the gesture-based interaction was more physically stressful since it requires the participant to move their finger in a larger 3D space instead of a small 2D surface, which could lead to fatigue over time.

- With the multi-modal interface, users felt distracted when trying to coordinate their two hands using different interfaces, requiring more time to think about which interface they should use during the different stages of tasks.

We have learned a few lessons from our study, and our results suggest that the following high-level guidelines for mobile AR interface designer with 3D gesture interaction:

- Do not use it for time consuming AR manipulation tasks because of the physical stress;

- The physical stress could easily lead to the mental stress eventually;

- Use it for more engaging experience like gaming instead of serious tasks requiring high accuracy;

- Try to enhance the depth perception by applying extra design or technology;

- Try to reduce the changing frequency of the interfaces in multi-modal to keep the user away from the distraction.

6.2 Future Directions

We presented three different frameworks that addressed three different levels of functionality and sophistication for hand gesture-based interaction - from the most simple 2D-based tracking, to 3D tracking of fingertips, to full 3D tracking of the whole hand skeleton. In the future, we would like to make comparison of the three related user studies that directly resulted from the studies done in this thesis, since we might be able to technically achieve high-quality 3D tracking of the full hand skeleton, there might always be situations where this is not possible – or even more important: where it might not be needed or even worse than a simpler implementation. We also see three more key future directions for research work in gesture-based interaction for mobile AR applications in a more general scope.

Full Hand Gesture Tracking Goes Mobile

The first and most straightforward direction is to improve the hand tracking technology and make 3D full-hand gesture estimation and tracking available on the mobile phone directly with the self-contained design. As discussed in Section 4.2, we prototyped the client-server framework to simulate full hand pose estimation on the mobile phone, although the system still required the server to work wirelessly, and was limited to a small working environment. In 2016 December, Leap Motion[1] announced the evolution of their hand tracking technology: the Leap Motion Mobile Platform. With this new reference design, they have cut down their software processing time by an order of magnitude while opening up higher performance and a massively expanded field of view, and aligned the Android SDK release with the arrival of Leap Motion-embedded mobile VR headsets. For the first time, full hand gesture tracking has become available on a smartphone directly in the self-contained mode with the help of their hardware module and software, as shown in Figure 6.1. This is the beginning of an important shift towards mobile and ubiquitous wearable displays that will eventually be as easy and casual to use as a pair of glasses. The ultimate result of this will be the merging of 3D free-hand gestures and physical realities anywhere and anytime.

Free-hand Gesture Supports for Remote AR Collaboration

A correlated research direction is to expand our free-hand gesture with the remote AR collaboration. With the growth of widespread fast data connections to the internet, the potential of remote collaboration has greatly increased [32]. In the real-world situations, remote expert guidance is always required by a local novice to complete physical tasks [1]. Traditional remote collaboration systems often use video cameras

[1] http://www.leapmotion.com/

FIGURE 6.1: Full-hand pose estimation on a smartphone with the Leap Motion sensor.

on each side to combine the view of both workspaces together and display the result on 2D monitors or mobile devices. In this case, the virtual guidance of the remote helper will be directly overlaid on the video of the worker's workspace to help the local worker finish the physical task [67] [2] [1] [32]. However, recent research shows that it is difficult for the remote helper to understand the spatial relationships between the objects in the local worker's workspace while using a 2D display [46] [108]. Furthermore, 2D remote guiding systems may not support complex hand gestures to enhance the efficiency of the collaboration.

Remote AR collaboration systems can enable visual aids and hand gestures used by a remote helper to be shared with a worker at a physical distance in 3D to assist him/her in performing manual tasks [108]. To explore this area, we have developed a prototype system that uses VR head mounted displays (HMDs) and depth sensors to capture the 3D surrounding environment of the local worker and map point-cloud data into the VR world, helping the remote helper understand the spatial relationships during the physical tasks. Hand gestures of the remote helper are also detected and shown in the 3D space to improve the communication.

In the worker space, the user performs a physical task by wearing a VR headset (Oculus Rift DK2[2]). In order to support the user to see the surrounding scene, one depth sensor (Soft Kinetic DS325) is attached to the front face of the headset, facing toward the workspace. The system can capture the current scene based on the aligned RGB frame and depth frame from the sensor, and map them into the VR world coordinate system as a dense point cloud. In the helper space, the user needs to wear the same device as well. The scene is captured and compressed on the worker side and wirelessly streamed to the helper side for him/her to watch and understand

[2] http://www.oculus.com/dk2/

the task and environment. Meanwhile, the depth sensor on the remote helper side detects the hand region and presents the hands as a point cloud colored based on different depth values (Figure 6.2).

FIGURE 6.2: The remote helper's hands are colored and overlaid on top of the local worker's hand to guide Lego assembling.

Both the acquired 3D point cloud of the current worker space and the hand gestures on the remote helper side are then fused together in real time, mapped into a single shared virtual coordinate system, and displayed in the VR headsets of both users. Furthermore, the orientation data from the built-in gyroscope sensor in the HMD can be accessed and shared wirelessly from the worker to the remote helper, and the remote side application will apply these data directly to the shared point-cloud view, so that the remote helper also needs to rotate their head with their headset to follow the local user's viewpoint. Finally, the remote helper can guide the worker to finish the task by using his/her own gestures such as finger pointing and movements, and the user in the worker space can understand what to do by following the hand gestures and voice of the remote helper.

Multi-modal Interfaces with Free-hand Gesture

We would also like to investigate the interfaces combining gesture with other interaction modalities, like speech recognition, eye tracking and physiological signal input, to evaluate the multimodal interfaces on handheld and wearable AR devices, and explore more complicated gesture input for various AR environment.

Appendix A

Questionnaires

This section presents the questionnaire used in the user study presented in Section 4.3, in which our proposed gesture-based interface within the client-server framework was compared to the traditional touch-based interface for canonical mobile AR manipulations.

PRE-QUESTIONNAIRE

[TESTING SEQUENCE: ☐ Touch first, Gesture second
☐ Gesture first, Touch second]

Age: _____

Gender:
☐ Male ☐ Female

Have you used Augmented Reality interface before?
☐ Yes ☐ No

Have you used Touch-based interface before?
☐ Yes ☐ No
If YES, then how often do you use it?
☐ Barely ☐ Sometimes ☐ Frequent

Have you used Hand Gesture-based interface before?
☐ Yes ☐ No
If YES, then how often do you use it?
☐ Barely ☐ Sometimes ☐ Frequent

Which hand do you use most for Touch input?
☐ Left ☐ Right ☐ Both

Which hand do you use most for Gesture input?
☐ Left ☐ Right ☐ Both

[TESTING CONDITION: ☐ Touch ☐ Gesture
☐ Translation ☐ Rotation ☐ Scaling]

I was performing WELL.

Totally Disagree			Neutral			Totally Agree
☐	☐	☐	☐	☐	☐	☐

The interface was EASY TO USE.

Totally Disagree			Neutral			Totally Agree
☐	☐	☐	☐	☐	☐	☐

The interface was EASY TO LEARN how to use.

Totally Disagree			Neutral			Totally Agree
☐	☐	☐	☐	☐	☐	☐

The interface was INTUITIVE; you feel straightforward to learn and use it without much thinking.

Totally Disagree			Neutral			Totally Agree
☐	☐	☐	☐	☐	☐	☐

The interface was NATURAL; you feel it similar to your daily interaction behaviours.

Totally Disagree			Neutral			Totally Agree
☐	☐	☐	☐	☐	☐	☐

The interface was USEFUL to do the task.

Totally Disagree			Neutral			Totally Agree
☐	☐	☐	☐	☐	☐	☐

The interface was MENTALLY STRESSFUL to use.

Totally Disagree			Neutral			Totally Agree
☐	☐	☐	☐	☐	☐	☐

The interface was PHYSICALLY STESSFUL to use.

Totally Disagree			Neutral			Totally Agree
☐	☐	☐	☐	☐	☐	☐

POST-QUESTIONNAIRE

Please select the interaction method that you prefer to use for different one-handed handheld mobile AR tasks:

Translation	☐ Touch	☐ Gesture	☐ Both
Rotation	☐ Touch	☐ Gesture	☐ Both
Scaling	☐ Touch	☐ Gesture	☐ Both

Please present the reason of your choice:

Please select the method that you prefer to use in general for one-handed handheld mobile AR interaction, and please present the reason of your choice:

☐ Touch ☐ Gesture ☐ Both

What advantages and problems are there for using hand gesture for one-handed handheld mobile AR interaction?

What advantages and problems are there for using touch input for one-handed handheld mobile AR interaction?

What task or applications would the hand gesture be useful or suitable for?

What needs to be improved for the hand gesture interaction method?

Any other comment?

Bibliography

[1] Leila Alem and Jane Li. "A Study of Gestures in a Video-mediated Collaborative Assembly Task". In: *Adv. in Hum.-Comp. Int.* 2011 (Jan. 2011), 1:1–1:7.

[2] Leila Alem, Franco Tecchia, and Weidong Huang. *HandsOnVideo: Towards a Gesture based Mobile AR System for Remote Collaboration*. New York, NY: Springer New York, 2011, pp. 135–148.

[3] Miroslav Andel et al. "Interactive Collaborative Scene Assembly Using AR on Mobile Phones". In: *Proceedings of the 16th International Conference on Advances in Artificial Reality and Tele-Existence*. ICAT'06. Hangzhou, China: Springer-Verlag, 2006, pp. 1008–1017.

[4] Mark Assad et al. "AR phone: Accessible Augmented Reality in the Intelligent Environment". In: *OZCHI2003*. 2003, pp. 26–28.

[5] Ronald T. Azuma. "A Survey of Augmented Reality". In: *Presence: Teleoper. Virtual Environ.* 6.4 (Aug. 1997), pp. 355–385.

[6] Huidong Bai, Lei Gao, and Mark Billinghurst. "Poster: Markerless Fingertip-based 3D Interaction for Handheld Augmented Reality in a Small Workspace". In: *Proceedings of the 8th IEEE International Symposium on 3D User Interfaces*. 3DUI '13. Orlando, FL, USA: IEEE Computer Society, Mar. 2013, pp. 129–130.

[7] Huidong Bai, Gun A. Lee, and Mark Billinghurst. "Freeze View Touch and Finger Gesture Based Interaction Methods for Handheld Augmented Reality Interfaces". In: *Proceedings of the 27th Conference on Image and Vision Computing New Zealand*. IVCNZ '12. Dunedin, New Zealand: ACM, Nov. 2012, pp. 126–131.

[8] Huidong Bai, Gun A. Lee, and Mark Billinghurst. "Using 3D Hand Gestures and Touch Input for Wearable AR Interaction". In: *Extended Abstracts on ACM CHI 2014 Conference on Human Factors in Computing Systems*. CHI EA '14. Toronto, Ontario, Canada: ACM, Apr. 2014, pp. 1321–1326.

[9] Huidong Bai, Gun Lee, and Mark Billinghurst. "Free-hand Gesture Interfaces for an Augmented Exhibition Podium". In: *Proceedings of the Annual Meeting of the Australian Special Interest Group for Computer Human Interaction*. OzCHI '15. Melbourne, VIC, Australia: ACM, 2015, pp. 182–186.

[10] Huidong Bai et al. "3D Gesture Interaction for Handheld Augmented Reality". In: *Proceedings of the 7th ACM SIGGRAPH Asia 2014 Symposium on Mobile Graphics and Interactive Applications*. SA MGIA '14. Shenzhen, China: ACM, Nov. 2014, 7:1–7:6.

[11] Huidong Bai et al. "Free-hand Interaction for Handheld Augmented Reality Using an RGB-depth Camera". In: *SIGGRAPH Asia 2013 Symposium on Mobile Graphics and Interactive Applications*. SA MGIA '13. Hong Kong: ACM, Nov. 2013, 22:1–22:4.

[12] Huidong Bai et al. "Work-in-progress: Markerless 3D Gesture-based Interaction for Handheld Augmented Reality Interfaces". In: *Proceedings of the 16th IEEE International Symposium on Mixed and Augmented Reality*. ISMAR '13. Adelaide, South Australia, Australia: IEEE Computer Society, Sept. 2013, pp. 1–6.

[13] Matthias Baldauf et al. "Markerless Visual Fingertip Detection for Natural Mobile Device Interaction". In: *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*. MobileHCI '11. Stockholm, Sweden: ACM, Sept. 2011, pp. 539–544.

[14] Fabio Bellavia, Domenico Tegolo, and Emanuele Trucco. "Improving SIFT-based Descriptors Stability to Rotations". In: *Proceedings of the 2010 20th International Conference on Pattern Recognition*. ICPR '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 3460–3463.

[15] Hrvoje Benko, Ricardo Jota, and Andrew Wilson. "MirageTable: Freehand Interaction on a Projected Augmented Reality Tabletop". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '12. Austin, Texas, USA: ACM, 2012, pp. 199–208.

[16] Mark Billinghurst, Tham Piumsomboon, and Huidong Bai. "Hands in Space: Gesture Interaction with Augmented-Reality Interfaces". In: *IEEE Computer Graphics and Applications* 34.1 (2014), pp. 77–80.

[17] Mark Billinghurst et al. *Developing Handheld Augmented Reality Interfaces*. New York, NY, USA: Oxford University Press, 2014, pp. 615–635.

[18] Doug A. Bowman et al. *3D User Interfaces: Theory and Practice*. Redwood City, CA, USA: Addison Wesley Longman Publishing Co., Inc., 2004.

[19] Michael Calonder et al. "BRIEF: binary robust independent elementary features". In: *Proceedings of the 11th European conference on Computer vision: Part IV*. ECCV'10. Heraklion, Crete, Greece: Springer-Verlag, 2010, pp. 778–792.

[20] D. Caruso, J. Engel, and D. Cremers. "Large-Scale Direct SLAM for Omnidirectional Cameras". In: *International Conference on Intelligent Robots and Systems (IROS)*. Sept. 2015.

[21] Sukmoon Chang. "Extracting Skeletons from Distance Maps". In: *International Journal of Computer Science and Network Security* 7.7 (July 2007), pp. 213–219.

[22] Lieu-Hen Chen, Chi-Jr Yu, and Shun-Chin Hsu. "A Remote Chinese Chess Game Using Mobile Phone Augmented Reality". In: *Proceedings of the 2008 International Conference on Advances in Computer Entertainment Technology*. ACE '08. Yokohama, Japan: ACM, Dec. 2008, pp. 284–287.

[23] Junyeong Choi et al. "Bare-hand-based augmented reality interface on mobile phone". In: *ISMAR*. 2011, pp. 275–276.

[24] Wendy H. Chun and Tobias Höllerer. "Real-time Hand Interaction for Augmented Reality on Mobile Phones". In: *Proceedings of the 18th International Conference on Intelligent User Interfaces*. IUI '13. Santa Monica, CA, USA: ACM, Mar. 2013, pp. 307–314.

[25] Adrian James Clark. "OPIRA: The Optical-flow Perspective Invariant Registration Augmentation and Other Improvements for Natural Feature Registration". PhD thesis. University of Canterbury, Christchurch, New Zealand, 2009.

[26] Andrew J. Davison. "Real-Time Simultaneous Localisation and Mapping with a Single Camera". In: *Proceedings of the Ninth IEEE International Conference on Computer Vision - Volume 2*. ICCV '03. Washington, DC, USA: IEEE Computer Society, 2003, pp. 1403–1410.

[27] J. Engel, T. Schöps, and D. Cremers. "LSD-SLAM: Large-Scale Direct Monocular SLAM". In: *European Conference on Computer Vision (ECCV)*. Sept. 2014.

[28] J. Engel, J. Stueckler, and D. Cremers. "Large-Scale Direct SLAM with Stereo Cameras". In: *International Conference on Intelligent Robots and Systems (IROS)*. Sept. 2015.

[29] Steven Feiner et al. “A Touring Machine: Prototyping 3D Mobile Augmented Reality Systems for Exploring the Urban Environment”. In: *Proceedings of the 1st IEEE International Symposium on Wearable Computers*. ISWC ’97. Cambridge, MA, USA: IEEE Computer Society, Oct. 1997, pp. 74–81.

[30] George W. Fitzmaurice. “Situated information spaces and spatially aware palmtop computers”. In: *Commun. ACM* 36.7 (July 1993), pp. 39–49.

[31] Lei Gao et al. “An Oriented Point-cloud View for MR Remote Collaboration”. In: *SIGGRAPH ASIA 2016 Mobile Graphics and Interactive Applications*. SA ’16. Macau, China: ACM, 2016, 8:1–8:4.

[32] Steffen Gauglitz et al. “Integrating the Physical Environment into Mobile Remote Collaboration”. In: *Proceedings of the 14th International Conference on Human-computer Interaction with Mobile Devices and Services*. MobileHCI ’12. San Francisco, California, USA: ACM, 2012, pp. 241–250.

[33] Juergen Gausemeier et al. “Development of a real time image based object recognition method for mobile AR-devices”. In: *Proceedings of the 2nd international conference on Computer graphics, virtual Reality, visualisation and interaction in Africa*. AFRIGRAPH ’03. Cape Town, South Africa: ACM, 2003, pp. 133–139.

[34] Sinem Guven, Steven Feiner, and Ohan Oda. “Mobile Augmented Reality interaction techniques for authoring situated media on-site”. In: *Proceedings of the 5th IEEE and ACM International Symposium on Mixed and Augmented Reality*. ISMAR ’06. Santa Barbara, CA, USA, Oct. 2006, pp. 235–236.

[35] Taejin Ha, S. Feiner, and Woontack Woo. “WeARHand: Head-worn, RGB-D camera-based, bare-hand user interface with visually enhanced depth perception”. In: *Mixed and Augmented Reality, 2014 IEEE International Symposium on*. ISMAR ’14. Munich, Germany: IEEE Computer Society, Sept. 2014, pp. 219–228.

[36] Taejin Ha and Woontack Woo. “ARWand: Phone-Based 3D Object Manipulation in Augmented Reality Environment”. In: *Ubiquitous Virtual Reality (ISUVR), 2011 International Symposium on*. July 2011, pp. 44–47.

[37] Chris Harrison, Hrvoje Benko, and Andrew D. Wilson. “OmniTouch: Wearable Multitouch Interaction Everywhere”. In: *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*. UIST ’11. Santa Barbara, CA, USA: ACM, Oct. 2011, pp. 441–450.

[38] Anders Henrysson and Mark Billinghurst. "Using a Mobile Phone for 6 DOF Mesh Editing". In: *Proceedings of the 8th ACM SIGCHI New Zealand Chapter's International Conference on Computer-human Interaction: Design Centered HCI*. CHINZ '07. Hamilton, New Zealand: ACM, July 2007, pp. 9–16.

[39] Anders Henrysson, Mark Billinghurst, and Mark Ollila. "Face to Face Collaborative AR on Mobile Phones". In: *Proceedings of the 4th IEEE/ACM International Symposium on Mixed and Augmented Reality*. ISMAR '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 80–89.

[40] Anders Henrysson, Joe Marshall, and Mark Billinghurst. "Experiments in 3D Interaction for Mobile Phone AR". In: *Proceedings of the 5th International Conference on Computer Graphics and Interactive Techniques in Australia and Southeast Asia*. GRAPHITE '07. Perth, Australia: ACM, Dec. 2007, pp. 187–194.

[41] Anders Henrysson and Mark Ollila. "UMAR: Ubiquitous Mobile Augmented Reality". In: *Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia*. MUM '04. College Park, Maryland: ACM, 2004, pp. 41–45.

[42] Anders Henrysson, Mark Ollila, and Mark Billinghurst. "Mobile phone based AR scene assembly". In: *Proceedings of the 4th international conference on Mobile and ubiquitous multimedia*. MUM '05. Christchurch, New Zealand: ACM, 2005, pp. 95–102.

[43] Ken Hinckley et al. "Two-handed Virtual Manipulation". In: *The ACM Transactions on Computer-Human Interaction* 5.3 (Sept. 1998), pp. 260–302.

[44] Tobias Höllerer and Steven Feiner. *Mobile Augmented Reality*. London, UK: Taylor and Francis Books Ltd., 2004.

[45] Tobias Höllerer et al. "Exploring MARS: Developing Indoor and Outdoor User Interfaces to a Mobile Augmented Reality System". In: *Computers and Graphics* 23 (1999), pp. 779–785.

[46] Weidong Huang and Leila Alem. "Supporting Hand Gestures in Mobile Remote Collaboration: A Usability Evaluation". In: *Proceedings of the 25th BCS Conference on Human-Computer Interaction*. BCS-HCI '11. Newcastle-upon-Tyne, United Kingdom: British Computer Society, 2011, pp. 211–216.

[47] Wolfgang Hürst and Joris Dekker. "Tracking-based Interaction for Object Creation in Mobile Augmented Reality". In: *Proceedings of the 21st ACM International Conference on Multimedia*. MM '13. Barcelona, Spain: ACM, 2013, pp. 93–102.

[48] Wolfgang Hürst and Casper van Wezel. "Gesture-based Interaction via Finger Tracking for Mobile Augmented Reality". In: *Multimedia Tools and Applications* 62.1 (Jan. 2013), pp. 233–258.

[49] Wolfgang Hürst and Casper van Wezel. "Gesture-based interaction via finger tracking for mobile augmented reality". In: *Multimedia Tools and Applications* (2012), pp. 1–26.

[50] Moon-Sub Jin and Jong-Il Park. "Interactive Mobile Augmented Reality system using a vibro-tactile pad". In: *VR Innovation (ISVRI), 2011 IEEE International Symposium on*. Mar. 2011, pp. 329–330.

[51] Simon Julier et al. "BARS: Battlefield Augmented Reality System". In: *In NATO Symposium on Information Processing Techniques for Military Systems*. 2000, pp. 9–11.

[52] Markus Kähäri and David J Murphy. "Mara: Sensor based augmented reality system for mobile imaging device". In: *2006 5th IEEEACM International Symposium on Mixed and Augmented Reality* (2006), pp. 180–180.

[53] Hirokazu Kato and Mark Billinghurst. "Marker Tracking and HMD Calibration for a Video-Based Augmented Reality Conferencing System". In: *Proceedings of the 2Nd IEEE and ACM International Workshop on Augmented Reality*. IWAR '99. Washington, DC, USA: IEEE Computer Society, 1999, pp. 85–93.

[54] Yan Ke and Rahul Sukthankar. "PCA-SIFT: a more distinctive representation for local image descriptors". In: *Proceedings of the 2004 IEEE computer society conference on Computer vision and pattern recognition*. CVPR'04. Washington, D.C., USA: IEEE Computer Society, 2004, pp. 506–513.

[55] David Kim et al. "Digits: Freehand 3D Interactions Anywhere Using a Wrist-worn Gloveless Sensor". In: *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*. UIST '12. Cambridge, MA, USA: ACM, Oct. 2012, pp. 167–176.

[56] Hyejin Kim, Gerhard Reitmayr, and Woontack Woo. "Interactive annotation on mobile phones for real and virtual space registration". In: *Proceedings of the 10th IEEE International Symposium on Mixed and Augmented Reality*. ISMAR '11. Basel, Switzerland: IEEE Computer Society, Oct. 2011, pp. 265–266.

[57] Georg Klein. "Visual Tracking for Augmented Reality". PhD thesis. University of Cambridge, 2006.

[58] Georg Klein and David Murray. "Parallel Tracking and Mapping for Small AR Workspaces". In: *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*. ISMAR '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 1–10.

[59] Georg Klein and David Murray. "Parallel Tracking and Mapping on a Camera Phone". In: *Proceedings of the 8th IEEE International Symposium on Mixed and Augmented Reality*. ISMAR '09. Orlando, FL, USA: IEEE Computer Society, Oct. 2009, pp. 83–86.

[60] Mathias Kölsch. "Vision Based Hand Gesture Interfaces for Wearable Computing and Virtual Environments". AAI3143800. PhD thesis. University of California, Santa Barbara, 2004.

[61] Mathias Kölsch and Matthew Turk. "Robust hand detection". In: *Sixth IEEE International Conference on Automatic Face and Gesture Recognition, 2004. Proceedings.* May 2004, pp. 614–619.

[62] Mathias Kölsch et al. "Multimodal Interaction with a Wearable Augmented Reality System". In: *Computer Graphics and Applications, IEEE* 26.3 (May 2006), pp. 62–71.

[63] Takeshi Kurata et al. "VizWear: Toward Human-Centered Interaction through Wearable Vision and Visualization". In: *Advances in Multimedia Information Processing - PCM 2001*. Lecture Notes in Computer Science 2195 (2001), pp. 40–47.

[64] Gordon Kurtenbach and William Buxton. "User learning and performance with marking menus". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: celebrating interdependence*. CHI '94. Boston, MA, USA, Apr. 1994, pp. 258–264.

[65] Daniel Kurz and Selim Benhimane. "Gravity-aware handheld Augmented Reality". In: *Proceedings of the 2011 10th IEEE International Symposium on Mixed and Augmented Reality*. ISMAR '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 111–120.

[66] Daniel Kurz and Selim Ben Himane. "Inertial sensor-aligned visual feature descriptors". In: *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. June 2011, pp. 161–166.

[67] Hideaki Kuzuoka. "Spatial Workspace Collaboration: A SharedView Video Support System for Remote Collaboration Capability". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '92. Monterey, California, USA: ACM, 1992, pp. 533–540.

[68] Tobias Langlotz et al. "Sketching Up the World: In Situ Authoring for Mobile Augmented Reality". In: *Personal Ubiquitous Computer* 16.6 (2012), pp. 623–630.

[69] Gun A. Lee, Huidong Bai, and Mark Billinghurst. "Automatic Zooming Interface for Tangible Augmented Reality Applications". In: *Proceedings of the 11th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry*. VRCAI '12. Singapore, Singapore: ACM, 2012, pp. 9–12.

[70] Gun A. Lee et al. "Freeze-set-go interaction method for handheld mobile Augmented Reality environments". In: *Proceedings of the 16th ACM Symposium on Virtual Reality Software and Technology*. VRST '09. Kyoto, Japan: ACM, Nov. 2009, pp. 143–146.

[71] Taehee Lee and Tobias Höllerer. "Handy AR: Markerless Inspection of Augmented Reality Objects Using Fingertip Tracking". In: *Proceedings of the 11th IEEE International Symposium on Wearable Computers*. ISWC '07. Boston, MA, USA: IEEE Computer Society, Oct. 2007, pp. 83–90.

[72] V. Lepetit, P. Lagger, and P. Fua. "Randomized trees for real-time keypoint recognition". In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. Vol. 2. June 2005, 775–781 vol. 2.

[73] Stefan Leutenegger, Margarita Chli, and Roland Y. Siegwart. "BRISK: Binary Robust invariant scalable keypoints". In: *Computer Vision, IEEE International Conference on* (), pp. 2548–2555.

[74] Can Liu et al. "Mobile Augmented Note-taking to Support Operating Physical Devices". In: *Mobile HCI 2011 Workshop on Mobile Augmented Reality*. Sept. 2011.

[75] David G. Lowe. "Distinctive Image Features from Scale-Invariant Keypoints". In: *Int. J. Comput. Vision* 60.2 (Nov. 2004), pp. 91–110.

[76] Elmar Mair et al. "Adaptive and generic corner detection based on the accelerated segment test". In: *Proceedings of the 11th European conference on Computer vision: Part II*. ECCV'10. Heraklion, Crete, Greece: Springer-Verlag, 2010, pp. 183–196.

[77] Anthony Martinet, Gery Casiez, and Laurent Grisoni. "Integrality and Separability of Multitouch Interaction Techniques in 3D Manipulation Tasks". In: *IEEE Transactions on Visualization and Computer Graphics* 18.3 (Mar. 2012), pp. 369–380.

[78] Krystian Mikolajczyk and Cordelia Schmid. "A Performance Evaluation of Local Descriptors". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 27.10 (Oct. 2005), pp. 1615–1630.

[79] Thomas B. Moeslund and Lau Norgaard. *A Brief Overview of Hand Gestures Used in Wearable Human Computer Interfaces*. 2003.

[80] Annette Mossel, Benjamin Venditti, and Hannes Kaufmann. "3DTouch and HOMER-S: Intuitive Manipulation Techniques for One-Handed Handheld Augmented Reality". In: *Proceedings of the 15th International Conference of Virtual Technologies*. VRIC '13. Laval, France: ACM, Mar. 2013, 12:1–12:10.

[81] Annette Mossel, Benjamin Venditti, and Hannes Kaufmann. "DrillSample: Precise Selection in Dense Handheld Augmented Reality Environments". In: *Proceedings of the Virtual Reality International Conference: Laval Virtual*. VRIC '13. Laval, France: ACM, 2013, 10:1–10:10.

[82] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós. "ORB-SLAM: A Versatile and Accurate Monocular SLAM System". In: *IEEE Transactions on Robotics* 31.5 (Oct. 2015), pp. 1147–1163.

[83] Raul Mur-Artal and Juan D. Tardós. "ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras". In: *CoRR* abs/1610.06475 (2016).

[84] Alaeddin Nassani et al. "Tag It!: AR Annotation Using Wearable Sensors". In: *SIGGRAPH Asia 2015 Mobile Graphics and Interactive Applications*. SA '15. Kobe, Japan: ACM, 2015, 12:1–12:4.

[85] Joseph Newman, David Ingram, and Andy Hopper. "Augmented Reality in a Wide Area Sentient Environment". In: *Proceedings of the IEEE and ACM International Symposium on Augmented Reality (ISAR'01)*. ISAR '01. Washington, DC, USA: IEEE Computer Society, 2001, pp. 77–86.

[86] Nobuyuki Otsu. "A Threshold Selection Method from Gray-Level Histograms". In: *IEEE Transactions on Systems, Man, and Cybernetics* 9.1 (Jan. 1979), pp. 62–66.

[87] Hyung Min Park, Seok Han Lee, and Jong Soo Choi. "Wearable Augmented Reality System Using Gaze Interaction". In: *Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*. ISMAR '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 175–176.

[88] Wayne Piekarski and Bruce H. Thomas. "The Tinmith System: Demonstrating New Techniques for Mobile Augmented Reality Modelling". In: *Proceedings of the 3rd Australasian Conference on User Interfaces - Volume 7*. AUIC '02. Melbourne, Victoria, Australia: Australian Computer Society, Inc., Jan. 2002, pp. 61–70.

[89] Thammathip Piumsomboon, Adrian Clark, and Mark Billinghurst. "Physically-based Interaction for Tabletop Augmented Reality Using a Depth-sensing Camera for Environment Mapping". In: *Proceedings of the 26th International Conference on Image and Vision Computing New Zealand*. IVCNZ '11. Auckland, New Zealand, Dec. 2011, pp. 161–166.

[90] Ram Rajesh, J Nagarjunan, and D Arunachalam. "Distance transform based hand gestures Recognition for power point presentation navigation". In: *Advanced Computing: an International Journal (ACIJ)* 3.3 (Jan. 2012), pp. 41–48.

[91] Jun Rekimoto. "GestureWrist and GesturePad: Unobtrusive Wearable Interaction Devices". In: *Proceedings of the 5th IEEE International Symposium on Wearable Computers*. ISWC '01. Washington, DC, USA: IEEE Computer Society, 2001, pp. 21–27.

[92] Jun Rekimoto. "Navicam: A Magnifying Glass Approach to Augmented Reality". In: *Presence: Teleoper. Virtual Environ.* 6.4 (Aug. 1997), pp. 399–412.

[93] Jun Rekimoto. "Transvision: A hand-held augmented reality system for collaborative design". In: *In Proceeding of Virtual Systems and Multimedia '96 VSMM '96 Gifu Japan 1820 Sept 1996*. Vol. 96. 1996, pp. 18–20.

[94] Damien Constantine Rompapas et al. "Dynamic Augmented Reality X-Ray on Google Glass". In: *SIGGRAPH Asia 2014 Mobile Graphics and Interactive Applications*. SA '14. Shenzhen, China: ACM, 2014, 20:1–20:1.

[95] Edward Rosten and Tom Drummond. "Machine learning for high-speed corner detection". In: *Proceedings of the 9th European conference on Computer Vision - Volume Part I*. ECCV'06. Graz, Austria: Springer-Verlag, 2006, pp. 430–443.

[96] Ethan Rublee et al. "ORB: An efficient alternative to SIFT or SURF". In: *Computer Vision (ICCV), 2011 IEEE International Conference on*. Nov. 2011, pp. 2564–2571.

[97] Dieter Schmalstieg et al. "The studierstube augmented reality project". In: *Presence: Teleoper. Virtual Environ.* 11.1 (Feb. 2002), pp. 33–54.

[98] Johannes Schöning et al. "Bimanual Interaction with Interscopic Multi-Touch Surfaces". In: *Proceedings of the 12th IFIP TC 13 International Conference on Human-Computer Interaction: Part II*. INTERACT '09. Uppsala, Sweden: Springer-Verlag, 2009, pp. 40–53.

[99] Byung-Kuk Seo et al. "One-handed Interaction with Augmented Virtual Objects on Mobile Devices". In: *Proceedings of the 7th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry*. VRCAI '08. Fusionopolis, Singapore: ACM, Dec. 2008, 8:1–8:6.

[100] Toby Sharp et al. "Accurate, Robust, and Flexible Real-time Hand Tracking". In: *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. CHI '15. Seoul, Republic of Korea: ACM, Apr. 2015, pp. 3633–3642.

[101] Gilles Simon. "In-Situ 3D Sketching Using a Video Camera as an Interaction and Tracking Device". In: *31st Annual Conference of the European Association for Computer Graphics - Eurographics 2010*. Norrköping, Sweden, May 2010.

[102] Rajinder S Sodhi et al. "BeThere: 3D mobile collaboration with spatial input". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2013, pp. 179–188.

[103] Thad Starner, Joshua Weaver, and Alex Pentland. "A Wearable Computer Based American Sign Language Recognizer". In: *Proceedings of the 1st IEEE International Symposium on Wearable Computers*. ISWC '97. Cambridge, MA, USA: IEEE Computer Society, Oct. 1997, pp. 130–137.

[104] Koh Sueda et al. "Micro AR for Education: Using Metaphors for Familiar Actions". In: *SIGGRAPH Asia 2011 Emerging Technologies*. SA '11. Hong Kong, China: ACM, 2011, 12:1–12:1.

[105] Ivan E. Sutherland. "A head-mounted three dimensional display". In: *Proceedings of the December 9-11, 1968, fall joint computer conference, part I*. AFIPS '68 (Fall, part I). San Francisco, California: ACM, 1968, pp. 757–764.

[106] Vildan Tanriverdi and Robert J. K. Jacob. "Interacting with Eye Movements in Virtual Environments". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '00. The Hague, The Netherlands: ACM, 2000, pp. 265–272.

[107] Simon Taylor, Edward Rosten, and Tom Drummond. "Robust Feature Matching in 2.3 μs". In: *Computer Vision and Pattern Recognition Workshops, 2009. IEEE Computer Society Conference on*. CVPRW '09. Miami, FL, USA: IEEE Computer Society, June 2009, pp. 15–22.

[108] Franco Tecchia, Leila Alem, and Weidong Huang. "3D Helping Hands: A Gesture Based MR System for Remote Collaboration". In: *Proceedings of the 11th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry*. VRCAI '12. Singapore, Singapore: ACM, 2012, pp. 323–328.

[109] Manuel Veit, Antonio Capobianco, and Dominique Bechmann. "Influence of Degrees of Freedom's Manipulation on Performances During Orientation Tasks in Virtual Reality Environments". In: *Proceedings of the 16th ACM Symposium on Virtual Reality Software and Technology*. VRST '09. Kyoto, Japan: ACM, 2009, pp. 51–58.

[110] Daniel Wagner and Dieter Schmalstieg. "Artoolkitplus for pose tracking on mobile devices". In: *Proceedings of 12th Computer Vision Winter Workshop CVWW07*. Ed. by M Grabner and H Grabner. 2007, pp. 139–146.

[111] Daniel Wagner and Dieter Schmalstieg. "First Steps Towards Handheld Augmented Reality". In: *Proceedings of the 7th IEEE International Symposium on Wearable Computers*. ISWC '03. Washington, DC, USA: IEEE Computer Society, Oct. 2003, pp. 127–135.

[112] Daniel Wagner et al. "Pose Tracking from Natural Features on Mobile Phones". In: *Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*. ISMAR '08. Cambridge, UK: IEEE Computer Society, Sept. 2008, pp. 125–134.

[113] Daniel Wagner et al. "Real-time panoramic mapping and tracking on mobile phones". In: *Virtual Reality Conference (VR), 2010 IEEE*. Mar. 2010, pp. 211–218.

[114] Daniel Wagner et al. "Towards massively multi-user augmented reality on handheld devices". In: *Proceedings of the Third international conference on Pervasive Computing*. PERVASIVE'05. Munich, Germany: Springer-Verlag, 2005, pp. 208–219.

[115] Jingtao Wang and John Canny. "TinyMotion: camera phone based interaction methods". In: *CHI '06 extended abstracts on Human factors in computing systems*. CHI EA '06. Montreal, Quebec, Canada: ACM, 2006, pp. 339–344.

[116] Robert Wang, Sylvain Paris, and Jovan Popović. "6D hands: Markerless Hand-tracking for Computer Aided Design". In: *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*. UIST '11. Santa Barbara, CA, USA: ACM, Oct. 2011, pp. 549–558.

[117] Cik Suhaimi Yusof et al. "A Review of 3D Gesture Interaction for Handheld Augmented Reality". In: *In Proceedings of the 3rd International Conference on Interactive Digital Media*. ICIDM '14. Sabah, Malaysia, 2014.

[118] T. Y. Zhang and C. Y. Suen. "A fast parallel algorithm for thinning digital patterns". In: *Communications of the ACM* 27.3 (Mar. 1984), pp. 236–239.